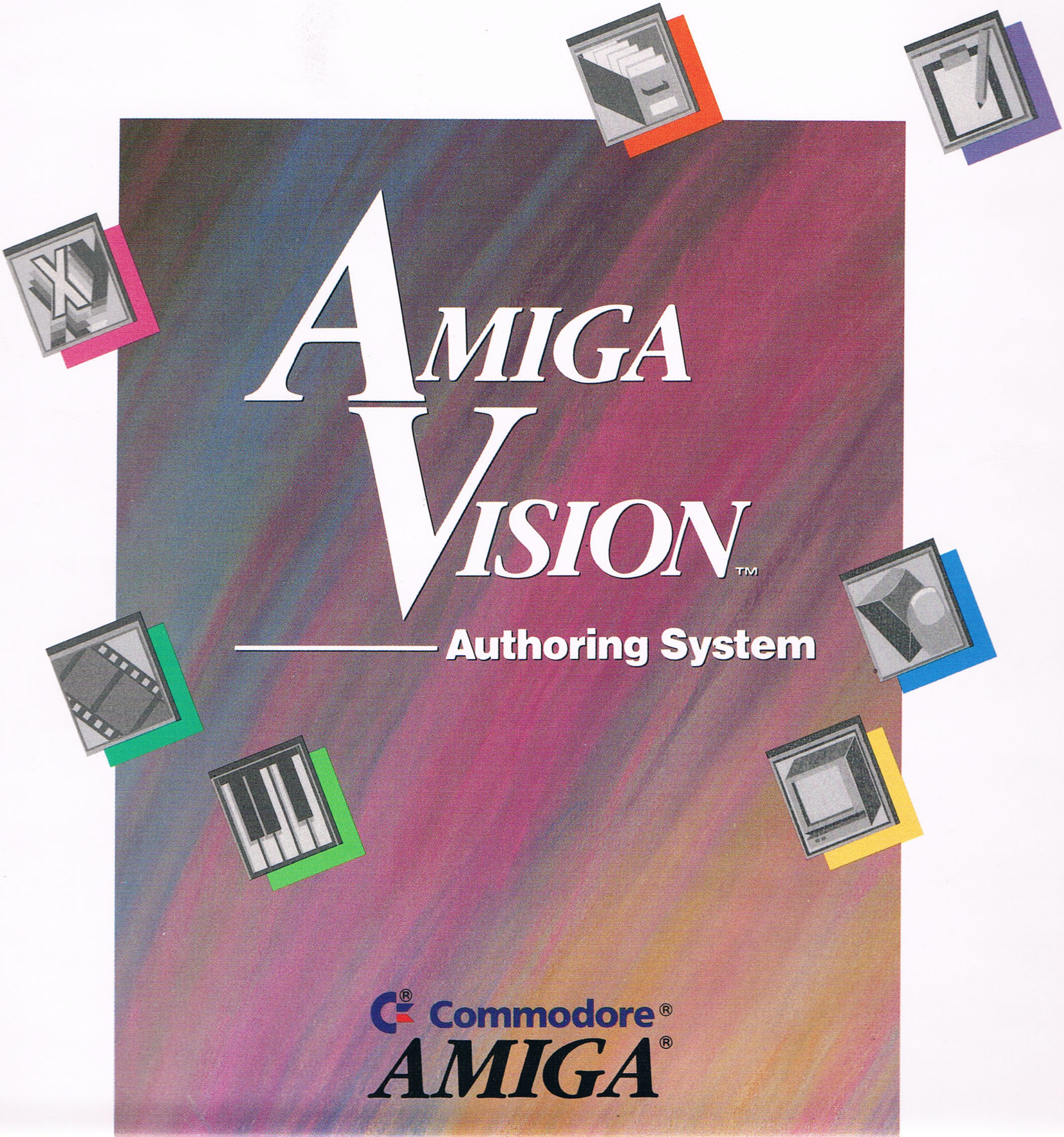
INTERACTIVE MULTIMEDIA SOFTWARE



AMIGA VISION™

Authoring System

Commodore®

AMIGA®

INTERACTIVE MULTIMEDIA SOFTWARE

AMIGA VISION™

Authoring System

Commodore®
AMIGA®

COPYRIGHT

DISCLAIMER

THE PROGRAM IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE PROGRAM IS ASSUMED BY YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU (AND NOT THE DEVELOPER OR COMMODORE ELECTRONICS LIMITED OR ITS DEALERS) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. FURTHER, COMMODORE ELECTRONICS LIMITED DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF, OR THE RESULTS OF THE USE OF, THE PROGRAM IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS, OR OTHERWISE; AND YOU RELY ON THE PROGRAM AND THE RESULTS SOLELY AT YOUR OWN RISK. IN NO EVENT WILL COMMODORE ELECTRONICS LIMITED BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE PROGRAM EVEN IF IT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME LAWS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITIES FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY.

Amiga is a registered trademark of Commodore Electronics Limited and Commodore. AmigaVision is a trademark of Commodore Electronics Limited and Commodore.

dBase is a registered trademark of Ashton-Tate Corporation.

Paradox is a registered trademark of Ausa Corporation.

Various third-party products referred to in this manual are trademarked by their owners and publishers. For a listing of products and their sources refer to Appendix E.

Printed in the U.S.A.

About This Manual

This manual is designed to let you start using AmigaVision with a minimum of effort. The manual begins with an introduction that summarizes AmigaVision's capabilities and features. This is followed by the two main parts of the manual: **User's Guide** and **User's Reference.** The **User's Guide** (Chapters 1 through 5) provides the basic information you need to use AmigaVision. The **User's Reference** (Chapter 6) provides comprehensive information you will turn to on a continuing basis.

The manual also includes five appendices, a glossary, and an index. Here's a brief description of each chapter and appendix:

Chapter 1 *Getting Started* , lists the items in the AmigaVision package, defines hardware and software requirements, explains how to create backup disks, and tells how to install AmigaVision.

Chapter 2, *Tutorial,* tells you what is on the two tutorial disks and takes you step-by-step through the elementary tasks involved in using AmigaVision.

Chapter 3, *How AmigaVision Works,* describes the use of the AmigaVision icons, file requesters, and pull-down menus.

Chapter 4, *Editors and Tools,* tells you how to create and modify display objects, variables and expressions, and database files and how to manipulate information contained on videodiscs.

Chapter 5, *Questions and Answers,* provides detailed answers to commonly asked questions about specific tasks.

Chapter 6, *Icons and Requesters,* lists and describes the purpose and use of each icon, together with a practical example. Also describes how requesters are used to define the specific action an icon performs.

Appendix A, *Error Messages,* lists AmigaVision error messages and tells you what to do when the messages appear.

Appendix B, *ARexx Interface,* explains how to use AmigaVision with ARexx, a new and powerful programming language. As an ARexx-compatible program, AmigaVision can exchange information with many other programs that use ARexx as a hub.

Appendix C, *Keyboard Shortcuts,* explains how to use a combination of two keystrokes to speed up editor operations.

Appendix D, *AAAE Video Player Device,* describes in detail the interface that C and Assembler programmers would use with the Ariadne video player device.

Appendix E, *List of Hardware and Software Companies*, provides a list of products that can be used with AmigaVision. The addresses of the companies selling the products are included.

NOTE
It is assumed that users of AmigaVision are familiar with standard Amiga operation as described in the documentation included with your computer.

ACKNOWLEDGEMENTS

AmigaVision was developed for Commodore Business Machines, Inc. by IMSATT Corporation. This product was the result of the collaborative effort, dedication, commitment and professionalism of the following individuals:

From IMSATT

Nat Kannan	*General Management, Design, and Documentation*
John Gerlach, Jr.	*Product Management, Design, and Software*
Mike Weiblen	*Design and Software*
Mike Crossmire	*Graphics Design and Documentation*
Wayne Lutz	*Software*
Joe Siebenmann	*Software*
Laura Roberts	*Quality Assurance*
Chris Rowley	*Documentation*

From Commodore Business Machines

John Campbell	*Project Director*
Cathy Godfrey	*Developer Support*
Karl Herrmann	*Product Management*
Jim Esch	*Documentation*
Gail Wellington	*Special Projects*
David Archambault	*Director of Business Markets*
Roy Strauss	*Multimedia Specialist*
Dan Baker	*Technical Assistance (music)*
Carolyn Scheppner	*Technical Assistance*

Consultants

Integrated Systems, Inc.

Chris Nicotra	*Design and Software*
Tom Brooks	*Software*

Ariadne Ltd. Inc.

Chris West	*Player.device*

Selfware, Inc.

Shane Thomas	*Software*

Contents

User's Guide

2. Tutorial

3. How AmigaVision Works

4. Editors and Tools

5. Questions and Answers

User's Reference

6. Icons and Requesters

Appendices

Welcome to AmigaVision

Product Overview

When we communicate with others, we use many ways to creatively convey information. Among these methods of communication are:

- Sound/Music
- Pictures
- Words
- Numbers
- Animated sequences
- Full-motion video

This is called the *multimedia* approach. A high school history teacher who shows a film or slideshow about the Civil Rights movement to his class is using the multimedia approach to supplement his normal lecture. Historically, multimedia presentations have been encumbered by the use of multiple technologies, such as slide projectors, videotapes, and computer graphics (remember how that high school teacher had trouble threading the film through the projector?). But today, powerful microcomputers offer a single delivery system for *integrated* multimedia presentations. Now the teacher needs to handle only one piece of equipment, the computer.

In addition to offering convenient integration of multimedia materials, microcomputers have made it possible to create *interactive* presentations, meaning that the viewer can actually participate by communicating with the computer. This interactivity has given rise to a whole new class of applications, called *courseware,* which are useful in teaching and training. Think back to that history class. You may have fallen asleep during the movie; but during an interactive presentation, you would be much more involved.

One computer in particular, the **Commodore Amiga**, has redefined multimedia presentation capabilities. A standard Amiga provides internal capabilities for speech synthesis, musical instrument sounds, specialized graphics functions and moving video images. Its multitasking operating system efficiently integrates these different capabilities.

AmigaVision lets you create multimedia presentations and courseware which combine graphs, charts, motion video, and sound.

Here's how it works:

- The audio and visual material for your presentation can be prepared using your favorite paint, animation, audio and text editing software.
- Using AmigaVision, you prepare a visual, icon-based outline of the logical flow of your course or presentation. Icons are little pictures on the computer screen that represent something else — a file, a program, a command.
- You then assemble the contents and sequence of the material by arranging icons in the visual outline. Some icons represent video, graphics, animation, sound, and text items. Others represent the flow and control elements of the presentation or course. AmigaVision uses the outline you create and additional information you supply to run your presentation.

AmigaVision supports the features of a typical high-level language like BASIC. Programming languages like BASIC and AREXX. These languages are text-based, which means that they use words as commands that the computer understands and acts upon. You then build programs through a sequence of command statements. AmigaVision, however, uses *icons* instead of words to control the substance and flow of your program.

If you have programmed with a text-based language before, you'll see that AmigaVision is a full-featured programming system in its own right, except that it's easier to use. If you have never programmed before, you'll see that AmigaVision lets you work intuitively because it is a visually oriented system. You'll be moving icons around on the screen, testing your programs as you go.

Intended Uses

The AmigaVision authoring system will allow you to create virtually any type of application: interactive business presentations, educational presentations, training programs, simulations, and even stand-alone applications, such as retail kiosks and instructional or training systems.

Business Presentations

Every day, businesses use presentations to sell products, services, ideas, and plans. The main objective of a business presentation is to persuade a customer by providing information in a precise and attractive manner and responding to questions promptly. AmigaVision lets you create such presentations as well as applications which allow group interactivity. With AmigaVision, you'll be prepared for those "What if?" questions.

Education

For schools, colleges, and corporate campuses, AmigaVision can be used in classroom presentations. As a teacher working with AmigaVision, you can quickly and easily prepare multimedia lectures and course materials. You can use examples randomly to suit the context of classroom discussion and illustrate complex concepts and ideas.

Training

The ability to train through Computer Based Training (CBT) is valuable to businesses because it often has significant time, safety, flexibility, and distribution advantages over other more traditional alternatives. Training can be changed or updated easily. CBT courses can be sent across the world, to locations where trainers may not be available.

Simulations

It is often valuable to simulate real world events to prepare novices for appropriate decision making. The computer, in combination with live video or animations, can replicate situations that would otherwise be infeasible because of cost or safety considerations. Novice nuclear engineers, for example, can be trained more safely and cheaply on simulations than on functioning nuclear reactors.

One-on-One Interactive

Interactive applications for AmigaVision include individualized teaching, training, and information kiosk applications. These differ from the other presentation applications in that the end user works one-on-one with the computer without personalized help. As a designer of these single-user applications, you need to anticipate how an end user might interact with your application, then incorporate interactions with the user, visual engagement for motivation, and mechanisms for continual feedback. AmigaVision gives you these features, allowing you to design flexible, highly interactive lessons and courseware.

Program Features

Audio Visual Display Elements

AmigaVision lets you select a variety of audio-visual elements to include in your project. The elements are:

- Computer graphics
- Computer animation
- Text in different fonts, sizes, and colors
- Digitized sound
- Music
- Synthesized speech
- Images and/or sound from a videodisc player

Using AmigaVision you can arrange these elements on the screen in the order of presentation, specify user input and responses and branch to the appropriate section of the program. You can immediately test and see the results and refine the flow of your presentation.

User Inputs

When you design an interactive presentation, demonstration, or course, you need ways for letting the end user respond to the computer. AmigaVision supports the following user input devices:

- Mouse
- Keyboard
- Touch screen

Joystick input is not supported.

A touch-screen behaves just like a mouse, except that you actually move the pointer around the screen with your finger. With AmigaVision you can define your application to support the mouse, and then the touch-screen driver will allow the user to press boxes on the screen and so forth.

For mouse and touch screen input, you can define "hit boxes" on the screen. A hit box is a region on the screen to which a specific key can be assigned. When a user points a mouse pointer and clicks inside this region (or touches the region with a finger), the corresponding key signal is returned. This is useful for menu selection, help requests, and so on. For example, in a given presentation, you might include two hit boxes on a screen—one titled "Continue", the other, "Quit". When the user clicks in a box, the corresponding action occurs.

For keyboard input, you may specify any alphabetic, numeric, or function keys as valid user input and also assign meanings to function keys. For instance, you might prompt the user to type "C" for Continue or "Q" for quit.

Control Structures

Control structures are the conditional *branching* and *looping* constructs that are necessary to define the flow of a program, presentation or course. You use these control structures to move around within your program, depending on different kinds of conditions. The available control structures are:

- If-Then
- If-Then-Else
- Branching
- Looping
- Calls to Subroutines

In AmigaVision, you use icons to specify control structures and determine how a program branches.

Editors

User inputs and control structures need *variables* and *expressions* for collecting and storing data and evaluating conditions. To create variables and expressions for conditional evaluation, you use the **Expression Editor**. All mathematical operators, functions, symbols, and logical operators that are commonly used in programming languages are supported by AmigaVision.

You can create a variety of graphical display objects, such as polygons and circles using AmigaVision's **Object Editor**. These objects can serve as hit boxes for mouse or touch screen input.

The Object Editor also allows you to specify simple text sequences as hit boxes with a choice of fonts and colors.

The **Videodisc Controller** allows you to choose video segments for inclusion in your project. You can view the video with or without frame numbers or audio. Using the controller, you can specify start and end frames, or search to a specific frame.

The **Database Editor** allows you to create, import, or modify an informational database for your presentation. You can design forms, create fields, and enter and edit data.

RunTime Module

Once you have completed a presentation or a course, the material can be prepared for use by others with the **RunTime** feature. This feature helps you to create the complete disks that you will distribute to end users.

User's Guide

Chapter 1

Getting Started

Chapter 1: Getting Started

Your AmigaVision package includes the following:

- AmigaVision Boot Disk
- AmigaVision Program Disk
- AmigaVision Tutorial Disk 1
- AmigaVision Tutorial Disk 2
- 3-ring binder
- AmigaVision User's Guide and Reference Guide
- Warranty Card

Hardware Requirements

Minimum

- Amiga Computer
- 1 megabyte of RAM

Recommended

- Amiga Computer
- 3 megabytes of RAM
- 20 megabyte hard drive

Hardware peripheral options include:

- A videodisc player for incorporating prerecorded video. AmigaVision supports the following players:
    - Pioneer 6000
    - Pioneer 4200
    - Pioneer 2200
    - Philips 835
    - Philips 410
    - Philips 405
    - Sony 2000
    - Sony 1550
    - Sony 1500
    - Sony 1200
    - Sony Umatic 9 tape player

- A touch-screen. This allows interaction without using the keyboard or mouse. Touch-screen monitors on the Amiga can behave as mouse emulators, allowing them to work with all Amiga applications. With AmigaVision, you can define your application to support the mouse, and then the touch-screen driver will allow the user to press hit boxes on the screen. We have included on the Tutorials disk a driver for the Elographics touchscreens.
- A Genlock. A genlock is a device that synchronizes computer graphics to a video signal and displays both graphics and video on a single monitor.
- Additional memory. Having more RAM (Random Access Memory) gives you greater flexibility in designing large and complex AmigaVision projects.
- A Hard Disk. The addition of a hard disk will allow you to store large and numerous files for use with AmigaVision. Your system will perform faster too. Note that Amiga 500 owners with the A590 Hard Disk should populate their A590 with 2 megabytes of additional memory in order to efficiently run AmigaVision. Otherwise, you may experience decreased performance.
- Additional floppy disk drive. If you are running AmigaVision from floppy disks, the addition of another drive will reduce the amount of disk swapping required on a single-drive system.

Software Requirements

AmigaVision requires AmigaDOS 1.3 or higher. You will need other software to create your own graphics, animation, sound and music. AmigaVision supports the standardized Amiga file formats adhered to by a wide variety of third-party products for creating graphics, animation and sound as shown below.

Type	Format
Graphics	ILBM
Music	SMUS
Digitized Sound	8SVX
Animation	ANIM5

Creating Backup Disks

AmigaVision is supplied on floppy disks. Before installing the software, make a backup copy of the original disks and store them in a safe place.

Warning: Never remove a floppy disk from the disk drive until the drive light goes out. Doing so may damage the data on the floppy.

Insert the "AmigaVision Boot" disk, and turn on your Amiga. After the Workbench has loaded, follow the steps below to make a backup of AmigaVision.

If your Amiga has two disk drives:

1. *Write-enable a blank floppy disk by closing the write protect notch in the corner of the disk. Insert the blank floppy disk in the second drive.*
2. *Select the "AmigaVision Boot" disk icon by clicking on the icon with the left mouse button, and dragging it on top of the blank disk icon.* This will make an exact duplicate of the disk.
3. *After the disk copy is completed, remove the original "AmigaVision Boot" disk from the first drive.*
4. *Select the "copy of AmigaVision Boot" disk icon and choose "Rename" from the Workbench menu. Use the cursor and delete keys to rename the disk "AmigaVision Boot" disk.*
5. *Place the "AmigaVision" disk in drive one and another write-enabled blank floppy disk in the second drive.*
6. *Select and drag the "AmigaVision" disk icon on top of the blank disk icon to make an exact duplicate of the disk.*
7. *When the disk copy is finished, remove the original "AmigaVision" disk from the first drive.*
8. *Select the "copy of AmigaVision" icon, choose Rename from the Workbench menu, and rename the disk "AmigaVision".*
9. *Store the original disks in a safe place.*

If your Amiga has only one disk drive:

1. *Insert the "AmigaVision Boot" disk into the disk drive.*
2. *Select the "AmigaVision Boot" disk icon and choose "Duplicate" from the Workbench menu.*
3. *Follow the instructions in the requesters to complete the copying process.*
4. *After the disk copy is completed, select the "copy of AmigaVision Boot" disk icon and choose "Rename" from the Workbench menu. Use the cursor and delete keys to rename the disk "AmigaVision Boot" disk.*
5. *Place the "AmigaVision" disk in the internal drive.*
6. *Choose "Duplicate" from the Workbench menu and follow the requesters to complete the copying process.*
7. *After the disk copy is completed, select the "copy of AmigaVision" disk icon and choose "Rename" from the Workbench menu. Rename the disk to "AmigaVision".*
8. *Store your original disks in a safe place.*

Remember to always use your working copy of AmigaVision, not the original.

Installing AmigaVision on a Hard Disk

If you are using Workbench:

1. *Insert the "AmigaVision" disk in a drive and open the disk by double-clicking on the disk icon.*
2. *Double click on the icon of the hard disk partition on which you wish to install AmigaVision.*
3. *Click on the Empty drawer and then select duplicate from the Workbench pull-down menu.*
4. *Rename the new drawer "AmigaVision" using the rename option from the Workbench pull-down menu.*
5. *Double click on the new AmigaVision drawer.*
6. *In the disk window, drag the AV icon to the hard disk partition you wish to install AmigaVision on. Then drag the "AV_Help" icon to the same hard disk partition.*

You need to have at least 800K of free space on that drive.

If you plan to use a videodisc with AmigaVision you will also need to perform the following steps:

1. *Insert the "AmigaVision Boot" disk in the internal drive.*
2. *Double click on your hard disk partition icon and double-click on the AmigaVision drawer.*
3. *In this drawer, double-click on the AV icon to start AmigaVision.*
4. *Select "Configuration" from the Project Menu to install the videodisc driver on your hard drive.*

5. ***Choose the appropriate videodisc from the list and click "OK."***

6. ***AmigaVision will tell you that it needs to create a directory called "Players" in your devices drawer to store the various videodisc drivers. Click "OK" to create the directory.***

   AmigaVision uses a device, Player.device, which must be present in the **DEVS** directory of your hard disk to operate a videodisc player.

7. ***AmigaVision will now ask you to locate the Player.device to copy to the hard disk. Click on "AmigaVision Boot" from the devices list, then "devs (dir)" from the file list, then click on the file "Player.device." Finally, click "OK" and AmigaVision installs the Player.device on your hard drive.***

   Choose your drawer from the list and click "OK" and AmigaVision copies the driver to the hard disk.

You may repeat the last step to copy as many of the drivers as you like. You will always need a copy of the "AmigaVision Boot" disk if you wish to install new videodisc drivers onto the system.

If You Are Using the CLI:

If you'd rather install AmigaVision and its videodisc drivers from the Command Line Interface instead of the Workbench, follow the instructions below. If you have already installed AmigaVision as described above, skip this section.

1. *Insert the "AmigaVision Boot" disk in drive DF0: (the Amiga's internal drive).*
2. *Open a command shell (by double clicking on the Shell icon in your system drawer).*
3. *At the prompt, type the following commands to install all the videodisc drivers into your DEVS drawer. Each line should be ended with a carriage return. You may choose to copy only the particular driver for your videodisc player.*

   ```
   makedir devs:players
   copy DF0:devs/player.device to devs:
   copy DF0:devs/players to devs:players
   ```

4. *After all disk activity has ceased, remove the "AmigaVision Boot" disk and insert the "AmigaVision Program" disk in drive DF0:.*
5. *Install the AmigaVision program on your hard drive with the following commands (The examples below use Work: as the hard drive volume name. Substitute the name of your hard drive when you type in these commands.):*

   ```
   makedir work:AmigaVision
   copy DF0:AmigaVision info to work:
   copy DF0:AmigaVision to work:AmigaVision all
   ```

   The "all" designator is essential to copying all of the AmigaVision support files, such as the Help files, onto the hard drive.

Starting AmigaVision

This section tells you how to start AmigaVision from floppy disks or a hard disk.

Floppy Disk Instructions

If you have two disk drives:

1. *Insert the "AmigaVision Boot" disk in the first drive (DF0:) and the "AmigaVision" disk in the second drive (DF1: or DF2:). Turn on your computer.*
2. *After Workbench has loaded, open the "AmigaVision" disk by double-clicking its disk icon.*
3. *Start AmigaVision by double-clicking the AV icon.*

If you only have one disk drive:

1. *Insert the "AmigaVision Boot" disk in the disk drive and turn on the machine.*
2. *After Workbench has loaded, remove the AmigaVision Boot Disk and insert the "AmigaVision" disk in the disk drive.*
3. *Open the "AmigaVision" disk by double-clicking its icon.*
4. *Start AmigaVision by double-clicking the AV icon.*
5. *Follow the instructions in the requesters to complete the loading of AmigaVision.*

Hard Disk Instructions

If you are using Workbench:

1. *Open the hard disk partition where you installed AmigaVision by double-clicking its icon.*
2. *Open the "AmigaVision" drawer by double-clicking its icon.*
3. *Start AmigaVision by double-clicking the AV icon.*

If you are using the CLI:

1. *Open a command shell by double-clicking on the Shell icon on your hard drive.*
2. *Enter the following command to change to the drive on which AmigaVision is installed.* (In this example, we will use Work: as the hard drive name. When you type the command, change this name to the real name of your hard disk.)

   ```
   cd work:AmigaVision
   ```

3. *Start AmigaVision by typing:*

   ```
   av
   ```

If you wish to start AmigaVision from another directory, you must specify the pathname. If AmigaVision is stored in the Work:AV directory, for example, type

```
work:AmigaVision/av
```

In order to multitask AmigaVision from the CLI, you must use the run command, as in

```
run work:AmigaVision/av
```

You may also affect the AmigaVision environment from the CLI by adding command line options. Command line options consist of single characters and must be separated by a space on the command line. For example, to start AmigaVision in interlace mode and turn on the double buffering option, type

```
av b d
```

To load the file MyDemo and automatically present it, type

```
av aMyDemo.AVf
```

Notice that there was no space between the **a** and the filename. There must be no space between the two for the file to be automatically presented. The command line options are each described below. They can appear in any order, but each must be separated by a space.

Description of Parameters

b Turns on automatic backups. Same as the BACK option.

d Turns on double-buffering. Same as the DBUF option.

i Runs AmigaVision on a high-resolution interlaced screen. Same as the LACE option.

m Tells AmigaVision to use as little memory as possible. Same as the LMEM option.

a Tells AmigaVision to load the subsequent filename and **Present . . .** that file. There must be no space between the a and the filename, and the path to the file must be specified.

Note: Although not absolutely necessary, we recommend that you ASSIGN AMIGAVISION: to the appropriate partition on your hard drive. For instance, if you have installed AmigaVision on WORK:AmigaVision, type the following:

```
ASSIGN AMIGAVISION: WORK:AMIGAVISION
```

Place this command in your Startup sequence so that the Amiga will always perform this operation.

Setting the AmigaVision Environment

The parameters described above may also be controlled through the standard Workbench Info requesters.

The Info requester can be displayed by selecting the AV icon from Workbench and selecting Info from the Workbench menu.

The field TOOLTYPES is where you can set AmigaVision's options. All options are defined on the same line separated by the vertical bar | character. AmigaVision's default is

OPTIONS = BACK

This setting means that AmigaVision will çreate backups, run on a medium resolution screen, assume plenty of free memory is available, not double-buffer displays, and **Present . . .** presentation icons when they are double-clicked from Workbench. To make AmigaVision run on a high-resolution interlaced screen and keep all other options the same, click in the TOOLTYPES field and change it to read

OPTIONS = BACK|LACE

The order of the options does not matter, but they must be separated with a | character. If you were to turn on every option, the line would read

OPTIONS = BACK|LACE|LMEM|DBUF|EDIT

If you leave any of the options out, that feature will not be used. Each option is described in detail below.

Option Descriptions

BACK This option instructs AmigaVision to create automatic backups of your files when you save. If you have a presentation called Aruba and you choose Save with the Back option ON, AmigaVision creates a copy of the old Aruba file and names it Aruba.bak for later retrieval. Future saves will replace Aruba.bak with the last version saved. Backups are desirable when more than one person is modifying a presentation and when there's a problem with your disks where a backup could be recovered. Backups are not desirable when you are short on disk space.

LACE AmigaVision normally runs in the Amiga's high resolution non-interlaced (640 x 200 pixels) mode. The LACE option forces AmigaVision into high resolution interlaced (640 x 400 pixels) mode, where you will be able to see twice as much data because all of the icons will be half as high. The LACE option is desirable when working with large presentations with a lot of logic where more icons on display make following the flow easier. The LACE option is not desirable in situations where memory is tight, as the high resolution interlace screen uses twice as much memory as the non-interlaced screen.

LMEM The low memory option tells AmigaVision to save as much memory as possible when presenting files. AmigaVision's Flow editor normally remains in memory and is quickly redisplayed when a presentation is over or is aborted. LMEM will cause AmigaVision to put the editor on hold and free the memory it is occupying, enabling more graphics, sounds and animations of larger sizes to be used. LMEM will cause AmigaVision to reload the editor when the presentation is done, which will add some time before returning to the editor. LMEM is desirable on Amigas with only one megabyte of RAM where all the memory may be needed in the presentation. LMEM is not desirable on Amigas with three megabytes of memory or more unless extremely large animations or digitized sounds are in use.

DBUF AmigaVision normally renders all of its graphics directly on the same video screen. Double buffering is a technique in which changes to a screen are rendered on a hidden screen that is only in memory while a different screen is being displayed. When all the new graphics are drawn, the new page is displayed. This technique offers the smoothest rendering of all graphics, animations, brushes and every other graphic operation. Double buffering requires more memory to handle a second screen of graphic memory and also can make some graphic rendering slower because of the time it takes to flip between the two screens in memory. DBUF is desirable when you feel you need the smoothest graphic rendering and transitions and have an accelerated Amiga. DBUF is not desirable on a standard Amiga or one with limited memory.

EDIT When you double-click an AmigaVision file from Workbench, it is normally loaded and automatically presented by AmigaVision and then you are returned to Workbench. The EDIT parameter tells AmigaVision to instead load the editor when you double-click files from Workbench. EDIT is desirable when you are still working on a project and you want to immediately jump into editing it. EDIT is not desirable when you have a finished project that you would just like to present.

Chapter 2

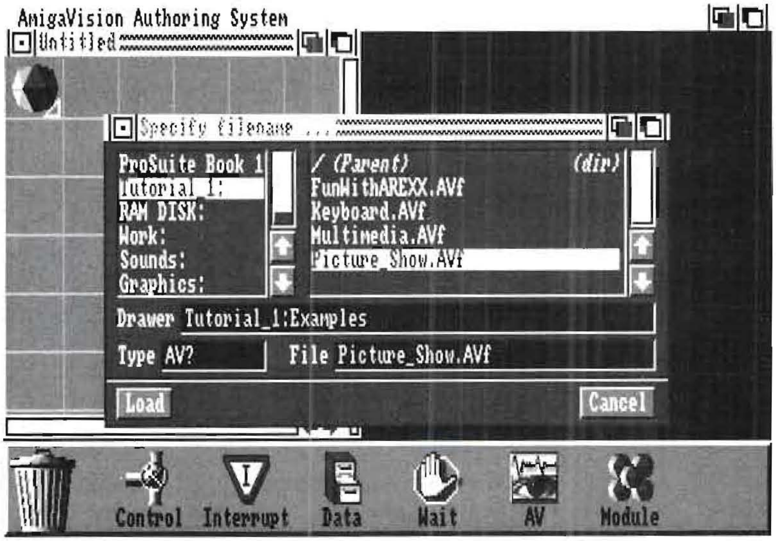
Tutorial

Chapter 2: Tutorial

Two tutorial disks are included with AmigaVision: Tutorial Disk 1 and Tutorial Disk 2. Disk 1 contains sample presentations called PictureShow, Multimedia, and Keyboard. Disk 2 contains Quiz and States. This chapter first demonstrates and then explains how each of these presentations works.

Picture_Show

1. *Load AmigaVision as described in Chapter 1. When the AV Authoring System screen appears, insert Tutorial Disk 1 into a floppy disk drive.*
2. *When the red light on the disk drive goes out, click and hold down the right mouse button, and move the mouse pointer to the top of the screen. Move the pointer to the left to the Project Menu. The Menu choices will pull down from the top of the screen.*
3. *Select "Load" from the Project Menu by holding down the right mouse button, moving the pointer to the "Load" option and releasing the right mouse button.*
4. *AmigaVision will open a requester titled Specify Filename. In this requester you choose the file you want to load. With the left mouse button, click once on the volume name Tutorial_1. When the wait pointer disappears, the right side of the requester window displays a list of subdirectories. Click the left mouse button once on Examples. When the wait pointer disappears, the requester window displays a series of filenames. Select the filename Picture_Show.AVf by clicking on it.*



AmigaVision Authoring System

Untitled

Specify filename ...

ProSuite Book 1
Tutorial_1:
RAM DISK:
Work:
Sounds:
Graphics:

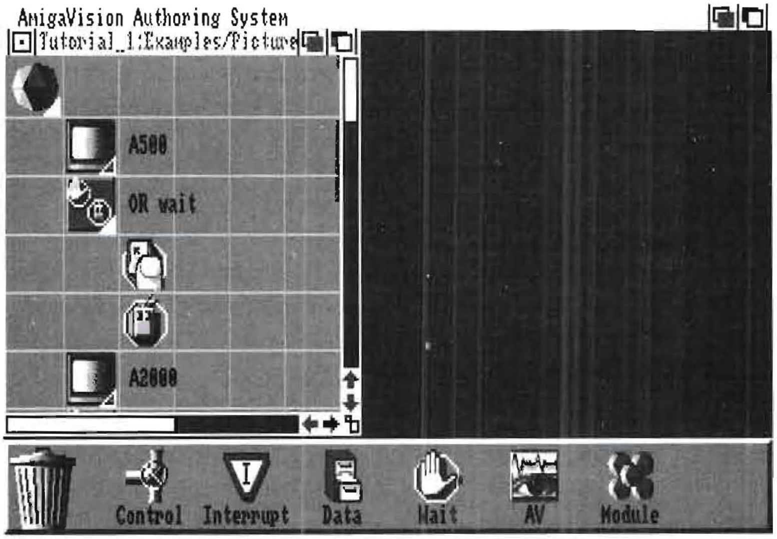
/ (Parent) (dir)
FunWithAREXX.AVf
Keyboard.AVf
Multimedia.AVf
Picture_Show.AVf

Drawer Tutorial_1:Examples

Type AV?

File Picture_Show.AVf

5. *After you have selected "Picture_Show.AVf", click on the Load gadget at the bottom left of the specify file name requester to load the file into AmigaVision. After the file is loaded, your screen will look like this.*



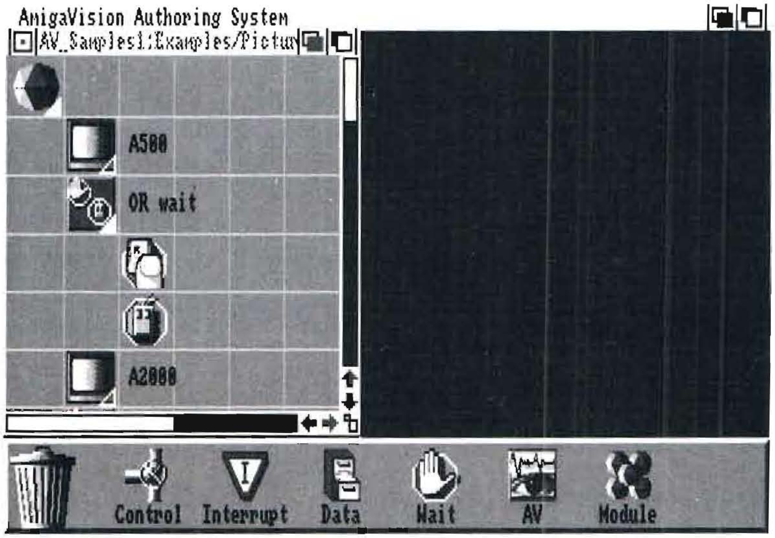
AmigaVision Authoring System

Tutorial_1:Examples/Picture

6. *Now let's play the presentation. Click and hold down the right mouse button and move to the Project menu, just as you did above. Now select "Present . . ." from the pull-down menu, and release the mouse button.*
7. *AmigaVision will display a sequence of pictures. To move from one picture to the next, click the left mouse button or press any key on the keyboard. AmigaVision will then display the next picture. Repeat these steps until the presentation ends. To stop the presentation, click on the right mouse button. You will be returned to the AmigaVision screen.*

How Picture_Show Works

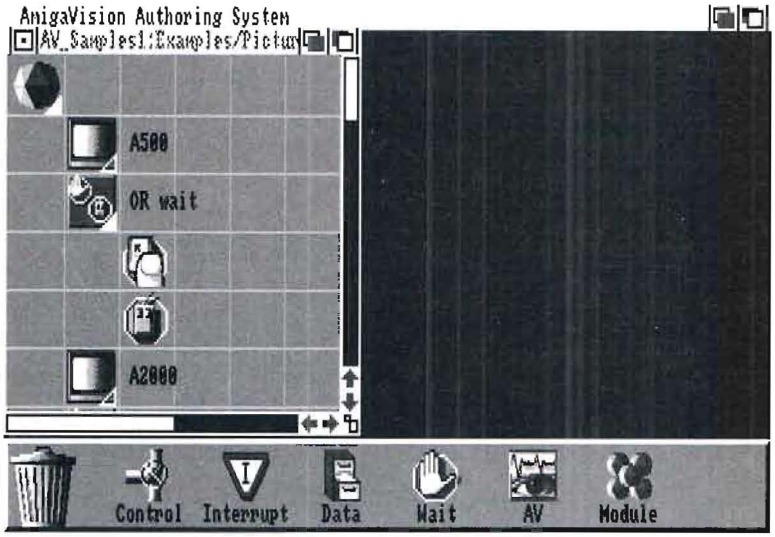
Now that you've seen the presentation, let's examine how it operates. At the bottom of your AmigaVision screen is a row of icons. This represents various submenus of icons that you use to build AmigaVision applications.



You'll also see a window with the name of your file in the title bar. This is called a Flow Window, and it contains the program you've just seen. As you can see, it is made up of different kinds of icons. AmigaVision starts at the top of this window and works from top to bottom as it proceeds through the presentation.

Many of the icons have names next to them. These are descriptive labels you can give to icons that help you distinguish them from one another.

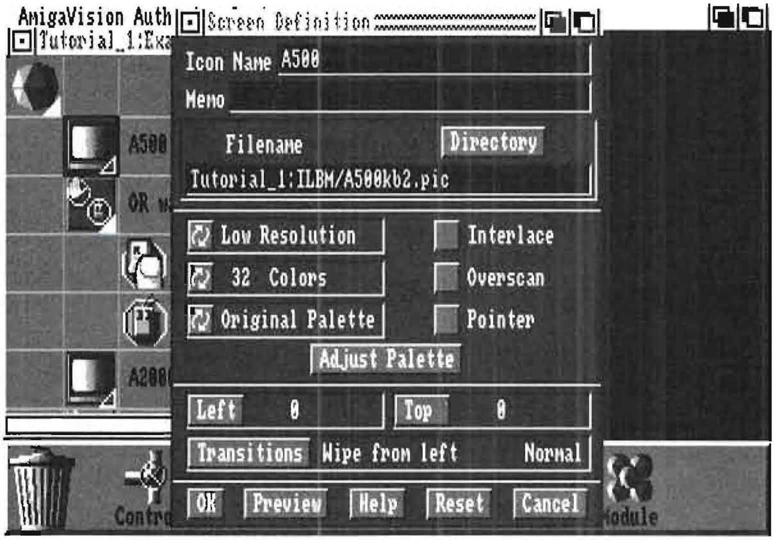
Let's start at the top of Example 1 and work downward. The first icon, which is unnamed on your screen, is called a **Module** icon. It essentially "contains" the rest of the presentation. We call it a parent icon. You can recognize a parent icon by the small triangle in the lower right corner of the icon.



Notice how all the other icons are not only below this module icon, but also to the right of it. When you place an icon below and to the right of a parent icon, that icon becomes the "child" of the parent icon. If you place an icon directly below another icon, those icons are siblings of one another. This presentation is made up of parents, siblings, and children. As a rule, AmigaVision will process icons in the following order. First, the instructions of the parent icon are performed, followed by those of its children. Then the instructions of the parent's siblings are executed, followed by those of its children, and so on. See Chapter 3 for a more complete explanation of icon relations.

Move down one step in the outline. To the lower right of the Module icon, you'll find a **Screen** icon. A **Screen** icon, when placed in your outline, will display a specified screen on your monitor. You can find the **Screen** icon in the **AV** submenu by clicking on the **AV** icon at the bottom of the AmigaVision screen. When you are creating a presentation, you place an icon in your outline by selecting it in the submenu, then dragging and placing the icon into the outline.

Double click on the **Screen** icon called A500 with the left mouse button. A **Screen Definition** requester will open. In this requester, you must define what file you want to display, designate the screen resolution, the number of colors used, the type of palette, the kind of transition, and so on. You define these operations by clicking on the various gadgets and typing in required information from the keyboard.



Screen Definition

Icon Name A500

Memo

Filename

Tutorial_1:ILBM/A500kb2.pic

Low Resolution

32 Colors

Original Palette

□ Interlace

□ Overscan

□ Pointer

Left 0

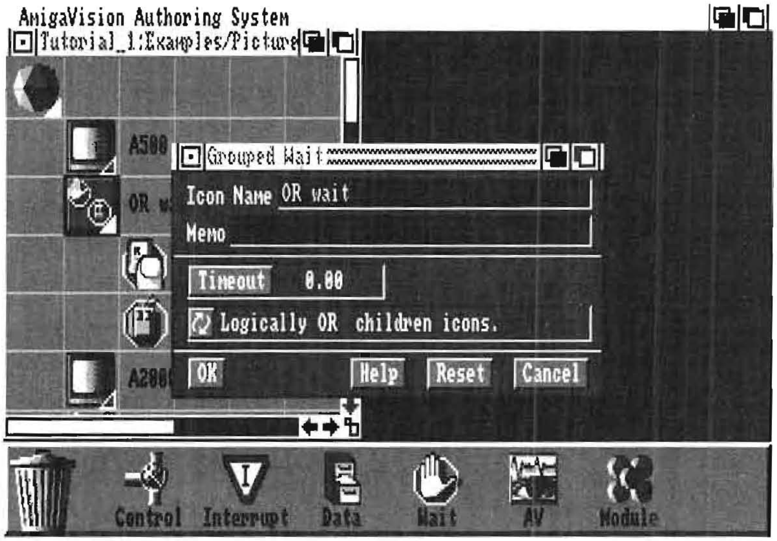
Top 0

Transitions Wipe from left Normal

For now, note the settings and then click on the Preview gadget. The actual screen will appear with the settings you have selected. Click on the right mouse button to return to the requester. If you aren't happy with those settings, you can change them and preview them again. Once you are satisfied with your settings, click on the OK gadget to return to the outline.

To summarize, the first screen icon in this outline has been defined to display a picture of an Amiga 500. Now AmigaVision will proceed to the next icon in the outline.

The next icon is called a **Wait** icon. Its function is to wait for a mouse click or a keyboard press before continuing the presentation. The **Wait** icon can be found by clicking on the **Wait** Submenu icon at the bottom of the screen.



AmigaVision Authoring System

Tutorial_1:Examples/Picture

Grouped Wait

Icon Name OR wait

Memo

Timeout 0.00

Logically OR children icons.

Note that the **Wait** icon in the Flow Window is a sibling of the icon above it and the parent of the two icons below and to the right of it. Double click on this icon. A Grouped Wait requester will open.

Here you can define how long AmigaVision will wait (Timeout) and also how it will understand the mouse and keyboard clicks that will follow. Click on the gadget with two arrows on it. You'll see another option. There are two options in this gadget:

1. ***Logically AND children icons.***
2. ***Logically OR children icons.***

What this means is that you have the option of waiting until all of the Wait icon's children have been performed before continuing or simply waiting until one of the children has been performed. This will make more sense when you see what comes after this icon.

There are two children of the **Wait** icon. The first is a **Keyboard** Icon, the second a **Mouse** Icon. Respectively, these icons wait for a keyboard press or a click of the Mouse button. After the designated key has been pressed, the presentation continues.

This presentation was designed to wait for either a keyboard press or a mouse click before continuing. If you had selected "Logically AND children's icons," as described above, AmigaVision would wait for *both* a key press *and* a mouse click before proceeding. Now click on Cancel in the Grouped Wait requester to return to the Flow Window.

The rest of this presentation is composed of the same building blocks that were used for the first segment. The only difference is that different filenames, screen resolutions, and transitions have been used. But the basic flow of the outline is as follows:

1. ***Display a screen.***
2. ***Wait for a key press or mouse click.***
3. ***Display the next screen.***
4. ***Wait for a key press or mouse click.***

And so on.

You may have noticed that the entire application does not fit into the window. To move around in this window and see the hidden elements below, use the sizing and scrolling gadgets.

You may want to experiment with this presentation, substitute your own picture files, or change the way the Wait icons work. Before you go on to Example 2, click on the Close Window gadget in the upper left corner of the Flow Window for Picture_Show. A requester appears, asking you to Save, Close or Cancel. Select close. If you want to record your changes, save the file under a different name so as not to change the original file.

Multimedia

Load the file Multimedia.AVf in the same way that you loaded the Picture_Show file. Then select **Present . . .** from the **Project** pull down menu. Make sure to have your sound turned up for this one.

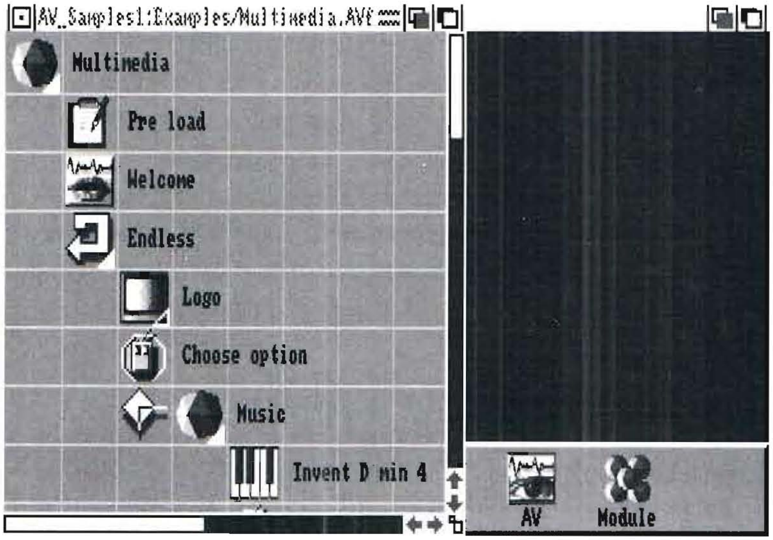
You will be given a welcome and presented with a menu screen. On this screen, click on the area you want to see demonstrated: Graphics, Music, Animation, Combination. Select Quit to exit the Presentation. If you want to halt the presentation before it completes, click the right mouse button.

How Multimedia Works

Let's take a look at the Flow Window and see how this application operates. The basic outline of the presentation is as follows:

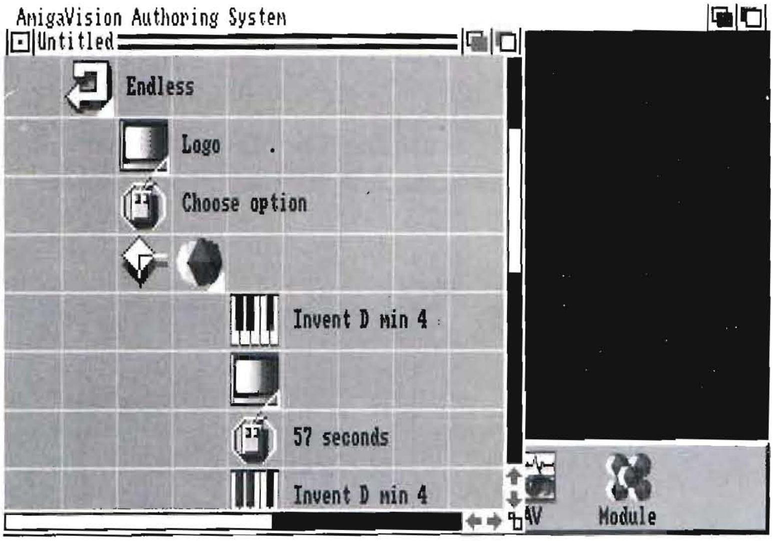
1. ***Say "Welcome to AmigaVision".***
2. ***Load a Screen with Menu Options.***
3. ***Wait for the user to choose an option.***
4. ***If user picks graphics, display a graphic.***
5. ***If user picks music, play music.***
6. ***If user picks animation, play animation.***
7. ***If user picks combination, play music and display graphics.***
8. ***If user picks Quit, exit the presentation.***

The first icon (after our parent module icon) is a **Resource** icon. You use the Resource icon to load various elements of your presentation into memory before the Presentation starts; then you don't have long delays in the middle of your presentation.



The next icon down is called a **Speak** icon. Double click on this icon to enter the Synthesized Speech requester. Here you can specify text that the Amiga will speak. You can also specify such elements as the gender of the speaker, the volume, pitch, and so on. In the presentation, this icon will say "Welcome to AmigaVision." Click on Cancel to close the Synthesized Speech requester.

Now as we proceed downward we encounter a **Loop** icon. Notice that it is a parent icon, and contains many children. A **Loop** means that after the last of the children has been performed, AmigaVision returns or "loops back" to the first child under the **Loop** icon.



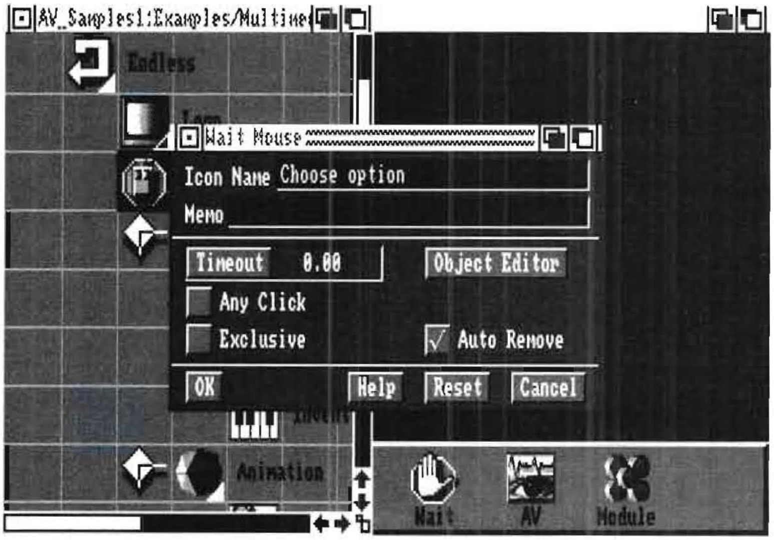
The children of the loop are as follows:

- A **Screen** icon, which displays the background screen for our menu.
- A **Wait Mouse** icon, which defines "hit boxes" (for entering user responses) on the screen, and waits for a mouse click that will return responses for AmigaVision to evaluate.
- Four **IF-ELSE** icons, which check to see which box was clicked on, and then perform actions based on the choice.

If you double-click on the **Loop** icon, you'll see that you can specify what kind of loop you want this to be. Click on the multistate gadget (the box with arrows in it) to see your different choices. For this example we want an endless loop. That means that if you choose to play music from the main menu, after the music ends, you'll be returned to the menu and be allowed to make another choice, until you select quit. Click on Cancel to close the Loop requester.

The **Screen** icon functions in the same way that the **Screen** icons in Picture_Show worked. They display an image on your monitor.

Note though how you can use the **Wait Mouse** icon. Double-click on the **Wait Mouse** icon. A Wait Mouse requester will open.



Icon Name Choose option

Memo

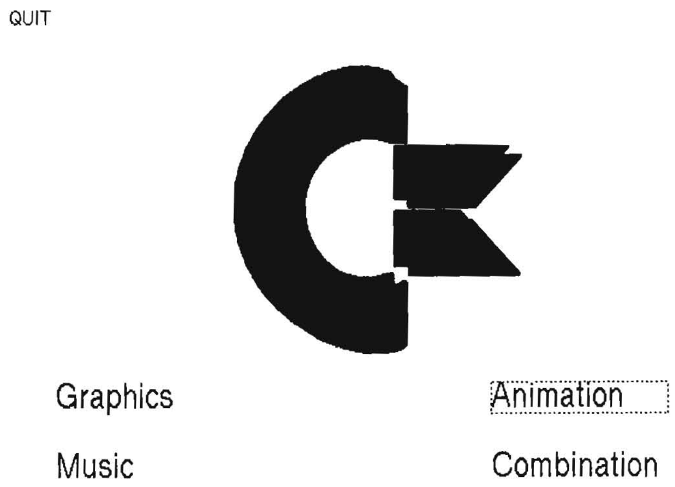
Timeout 0.00

□ Any Click

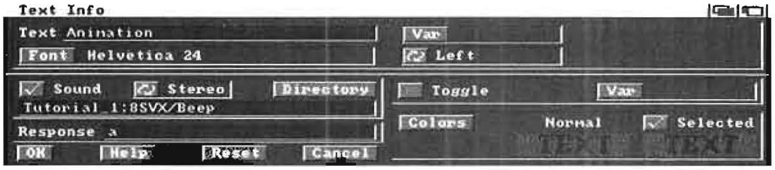
□ Exclusive

☑ Auto Remove

Now click on the Object Editor gadget. A new screen will open. Click on OK to display the background image.



You have entered the Object Editor. Here you can define various shapes that can be defined as hit boxes (areas of the screen which, when clicked on, mean something to AmigaVision). In this example the hit boxes have already been created. Click once on any of the menu choices. Notice how the word was surrounded by a dotted line. This is the border of the hit box. Double-click on this hit box now. An Info Requester will open.



Text Info

Text Animation

Font Helvetica 24

Left

☑ Sound

Stereo

Tutorial_1:8SVX/Beep

Response a

□ Toggle

Normal

☑ Selected

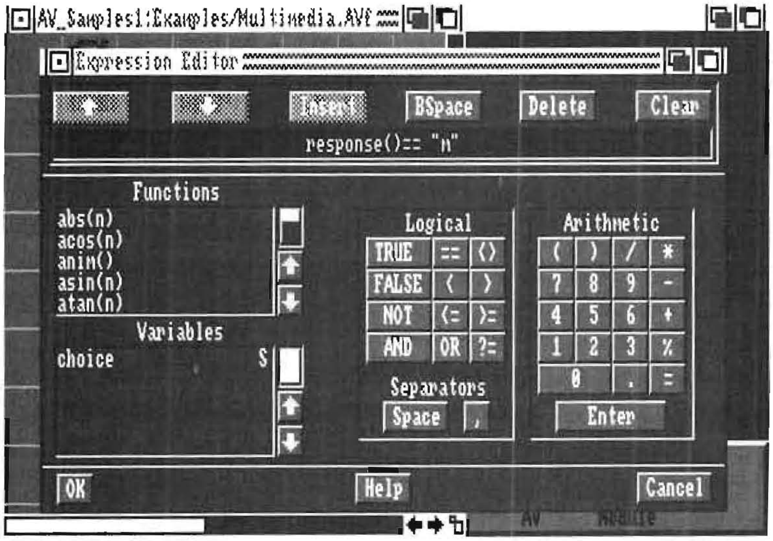
In this Requester you can define various attributes for your object (box). Most importantly for this example, notice the Response text field. For each hit box, a different Response String has been entered.

What does all this mean? It means that when you click on that box, AmigaVision will record a response. For example, if during the presentation you click on Animation, AmigaVision will return the Response "a". It is then later evaluated in an IF-ELSE expression and acted upon. Examine the Info Requesters for all the choices carefully so that you understand how they are defined. Note also that you can specify feedback sound and color for your object. In this case, when you click on a box, you'll hear a beep.

Select CANCEL to exit the Info Requester. Then exit from the Object Editor by pressing and holding the right mouse button, moving to the top left of the display, and selecting Exit from the Project pull-down menu. You will be returned to the Wait Mouse Requester. Click on Cancel to return to the presentation outline.

Now comes the point in our outline where the responses must be evaluated and acted upon. At this stage in the presentation, the user has clicked on one of the choices. We need to check and see which box was clicked.

The next icon you encounter displayed to the left of a module icon is an **IF-ELSE** icon. Double-click on this icon, and you'll enter the Expression Editor.



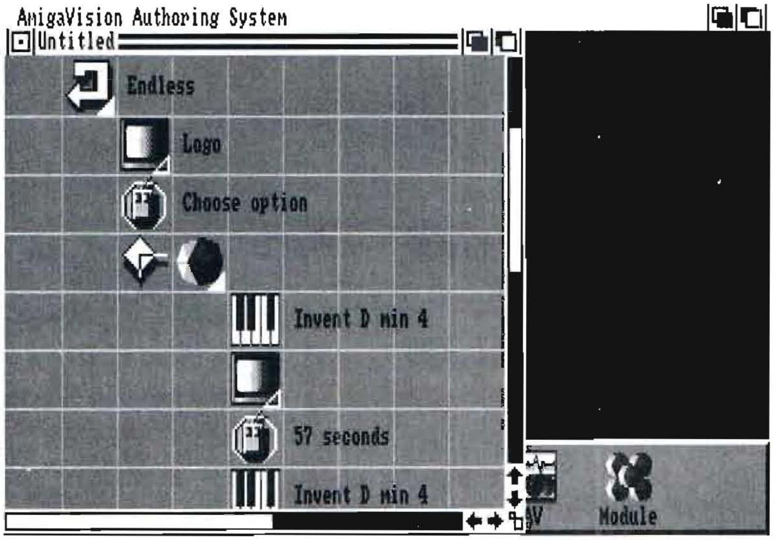
response()== "m"

Here, the response that was recorded by the **Wait Mouse** icon is evaluated in the expression **response() = = "m"** (this is found in the text box near the top of the expression editor window).

This expression means that AmigaVision checks for a response. If that response is "m" for music (remember that the Response string in the Music hit box was defined as "m") then the Expression is True and AmigaVision will proceed with the icon *next to* the **IF-ELSE** icon. If the expression is not true (that is, if Music was not clicked on) then the expression is false, and AmigaVision continues with the next sibling (the next icon directly below) of the **IF-ELSE** icon.

Each of the **IF-ELSE** icons in this example acts in the same way; it just checks for different response strings.

If, in this first example, the user clicks on the Music hit box, then our first expression is true, and AmigaVision proceeds to the **Module** Icon right next to the **IF-ELSE** icon. This **Module** icon contains the children necessary to play a Music file. The **Music** icon (which looks like a piano keyboard) works in much the same fashion as a **Screen** icon, except that it plays a music file. The first **Music** icon Starts the music, the **Wait Mouse** icon waits for a specified amount of time (57 seconds) or a click of the mouse button, and the second **Music** icon stops the music if the mouse has been clicked.



You should begin to see how all of the **IF-ELSE** constructions in this example are similar. They check for a condition (what the response was) then proceed in a direction based on whether the expression is TRUE or FALSE.

So the next **IF-ELSE** icon checks for the Animation response "a". If TRUE, AmigaVision plays an Animation, then waits for 22 seconds or a click of the mouse button.

The next **IF-ELSE** icon checks for the Graphics response "g". If TRUE, AmigaVision displays a picture, then waits for 10 seconds or a click of the mouse button before continuing to the next picture, which is also displayed for 10 seconds. The GFX icon causes the picture to color cycle, or change some of the colors in a preset range.

The next **IF-ELSE** icon checks for the Combination response "c". If TRUE, AmigaVision plays music and displays pictures in Combination. Examine closely the **Music** icon in this section. Double click on the icon to open the Music Requester, and look for the gadget next to the word Pause. If you click on this box, a check mark will appear. This means that AmigaVision will pause and wait for the music to end before proceeding. If left unchecked, as is the case in this example, AmigaVision will begin playing the music then continue on to the next icon, which displays a picture. **Wait Mouse** icons have also been included in this segment to give the user the ability to abort the demonstration and return to the menu. Click on Cancel to return to the Flow Window.

The last **IF-ELSE** icon checks for the Quit response "q". If TRUE, Amiga says "Goodbye" (using Synthesized speech) and then quits (using the **Quit** icon).

The presentation will endlessly loop after a demonstration until the quit response has been selected.

Before you go on to Quiz, click on the Close Window gadget in the upper left corner of the Flow Window.

Keyboard

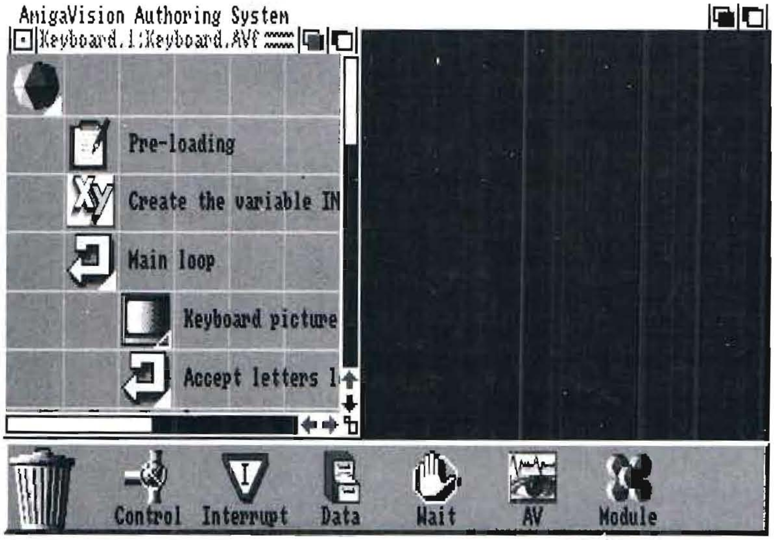
Load the file Keyboard.AVf and select **Present . . .** from the **Project** pull down menu. Make sure to have the volume turned up.

A screen will appear which resembles a typewriter keyboard. Enter your name. If you have a touch screen, you simply touch the keys on the screen keyboard. If you have a mouse, point and click on the letters of your name. Note that what you type appears at the top of the screen. Use the backspace key (the key with the left arrow at the bottom right of the keyboard) if you make a mistake. Then select the Done key when you are finished. Your Amiga will speak the name which was entered into the keyboard and clear the display area. You may repeat this process with different words as many times as you'd like. To exit the program type either the word "QUIT" or "EXIT" at the keyboard and select Done.

How Keyboard Works

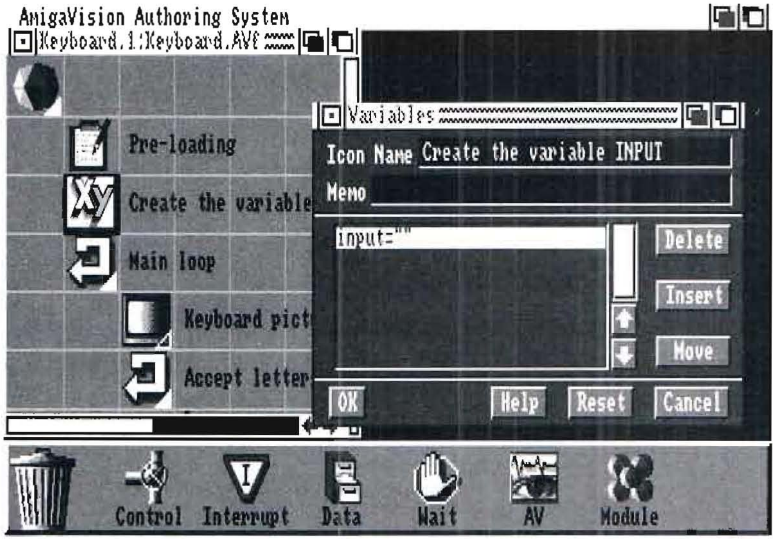
Let's take a look at the Flow Window and see how this application operates. The basic outline of the presentation could be described as follows:

1. ***Load the keyboard screen.***
2. ***Wait for the user to enter letters and to select the Done key.***
3. ***Say the message.***
4. ***Clear the text entry area and loop back to the beginning.***



The first icon (after the parent module icon) is a **Resource** icon. As you saw in the Multimedia example, you use the Resource icon to load various elements of your presentation into memory before the Presentation starts, so you don't have such long waits in the middle of your presentation. In this case, the screen and brushes which constitute the final picture of the keyboard are loaded in advance.

The next icon down is a **Variable** icon labeled "Create the variable INPUT". Double click on it to see how the variable was set. The expression **input = ""** establishes that the variable "input" will be text. Eventually, it will represent the word the user enters into the keyboard. AmigaVision will now recognize the variable called INPUT throughout the rest of the program. Click on Cancel to close the Variables requester.

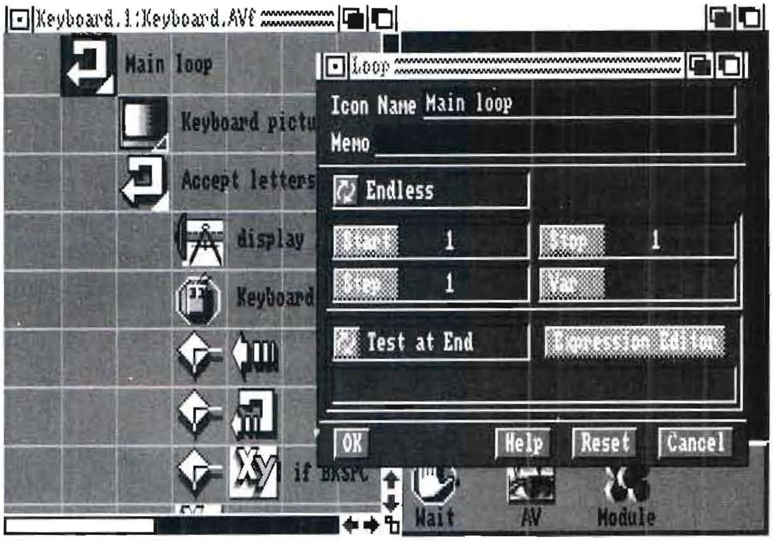


Icon Name Create the variable INPUT

Memo

input=""

Under this icon, you will find the **Loop** icon. The Loop icon is used to go through the children of the loop and then cycle back to the beginning of the loop again. Double-click on the Loop icon to see the Loop requester. Notice that this loop has been set up as "Endless". The cycling process will run endlessly. Select Cancel to exit this requester.



Loop

Icon Name Main loop

Memo

☑ Endless

Start 1 Stop 1

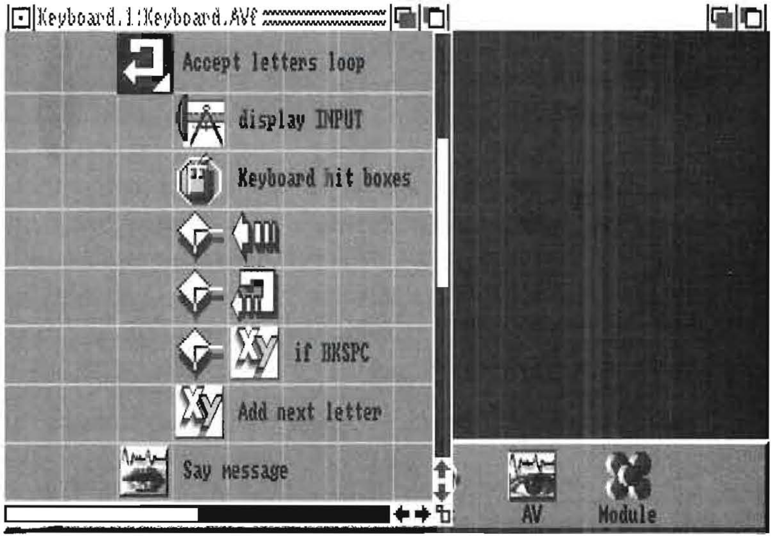
Step 1 Var

☑ Test at End

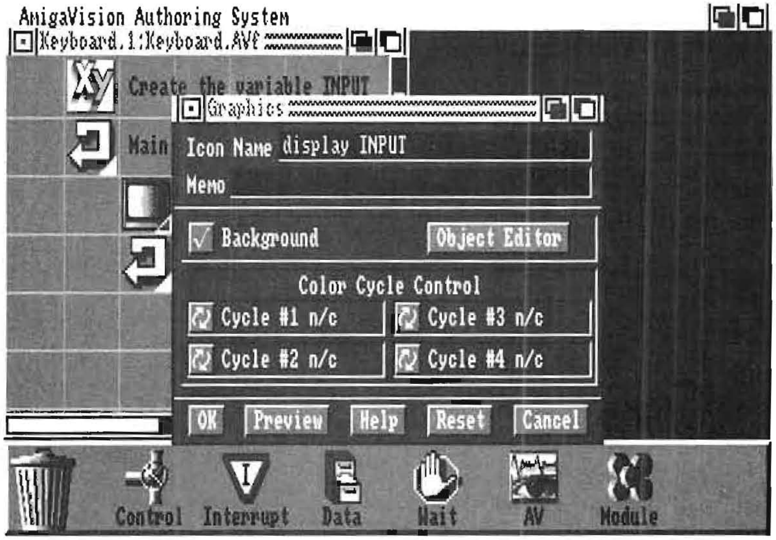
The first child of the loop (labeled "Keyboard picture") is the **Screen** icon. This icon is used to display the picture of the keyboard, which was created in a separate paint program and imported into AmigaVision. Separate brushes were also loaded into AmigaVision so each key is highlighted in a different color when you click on it.

Double-click on this Screen icon to see the Screen Definition requester. This requester was used to shape the way the image appears for the user. This picture was defined as a high resolution, 8-color picture using the original palette of colors used in the paint program. Notice the checkmark on the Pointer gadget. This indicates that a pointer will appear on the screen for the user. Exit this requester by selecting the Cancel gadget.

Under the Screen icon you will find another **Loop** icon (labeled "Accept letters loop"). You may wonder why another loop is necessary when you already have one at the beginning of the process. The second Loop icon is necessary because adding each individual letter is a separate process. In other words, the children of the second Loop icon represent the process of adding just *one* letter. It cycles through as each letter is added, until the user selects "Done." It then goes to any remaining children of the *first* Loop and cycles back to the beginning so the process begins again. Click Cancel to exit this requester.



The **GFx** (or **Graphics**) icon is under the "Accepts letter loop" icon (labeled "Display INPUT"). **GFx** icons are found under the AV submenu and are used to display text and objects which have been created in the object editor. In this example, we use one to display the text the user has entered into his on-screen keyboard (remember that this text was defined by the variable INPUT). Double click on the GFx icon to see the Graphics requester. The checkmark next to Background means the text will be "stamped on the background"—meaning it will not flicker each time a new letter is added (as it would if Background were not selected). The GFx icon may also be used to turn on "color cycling" in the gadgets labeled Cycle #1 through Cycle #4. Color cycling is a way to produce an "animated" effect on a screen—the colors on the screen change and thus produce the effect of motion. This effect was not used in this example; the gadgets read "n/c" to specify "no change".



AmigaVision Authoring System

Keyboard.1:Keyboard.AVf

Create the variable INPUT

Main

Graphics

Icon Name display INPUT

Memo

☑ Background

Color Cycle Control

Cycle #1 n/c

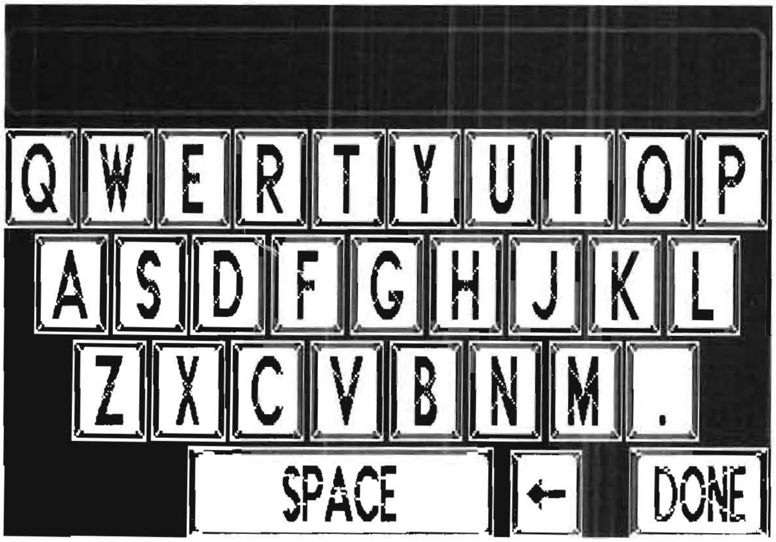
Cycle #3 n/c

Cycle #2 n/c

Cycle #4 n/c

Now select the Object Editor gadget. The large display of the word "input" represents *where* the user's response will be displayed on the screen. Its hit box was specifically placed in this area to correspond with the display area on the keyboard. A requester will appear, asking you if you want to load the picture of the keyboard in the background. Select OK. Now double-click on the word "input" to see its Info requester. This is where the font (Helvetica 24) was set. Notice in the requester's upper right corner that the variable for this hit box was defined as "input". This is how AmigaVision knows to place the user's response in this box. Select Cancel to leave the Graphics requester and Exit from the Project pull-down menu to leave the Object Editor. Select Cancel from the Graphics requester.

The second child of the loop is the **Wait Mouse** icon (labeled "Keyboard hit boxes"). Double-click on it to see its Wait Mouse requester and then select the Object Editor gadget. A requester will appear, asking you if you want to load the keyboard picture in the background. Select Cancel.



On the Object Editor screen, you will see boxes which represent the keys on the keyboard. These boxes were created with the Objects menu item called "Brush." They function as hit boxes and also mark where the brushes will be loaded into the picture. As mentioned earlier, the brushes for the keyboard were added so the outline of a key is highlighted when the user selects it. Double click on a box to see its Brush Info requester. Normally brushes appear surrounded by a rectangular box. The checkmark next to Cookie Cut means that the brushes will appear with their backgrounds "cookie cut" out of the picture—hence, no rectangular box behind them. The Response field is to correspond to the key on the keyboard—thus, RESPONSE equals the individual letter that the user entered on the keyboard. The Selected field displays the brush which will appear when that key has been *selected* and contains the name of the brush file which was loaded into the picture. Select Cancel to exit this requester, and select Exit from the Project menu. Next, select Cancel to exit the Wait Mouse requester.

Now that the keyboard picture is ready, with hit boxes and corresponding letters fully set, the user enters a letter. The next three **If-Else** statements refer to exiting the program and two special keys on the keyboard—the "Done" key and the "Backspace" key (the left arrow key located above Enter).

Double-click on the first **If-Else** icon. The Expression Editor displays expression **input = = "QUIT" OR input = = "EXIT"** in its display area. This expression means if the user enters either the word "quit" or "exit" on the keyboard the expression is true and AmigaVision proceeds right in the Flow Window. If the user enters anything other than "QUIT" or "EXIT", AmigaVision proceeds downward to the next If-Else icon. Select Cancel to exit the Expression Editor.

To the right of the first If-Else icon is the **Quit** icon. If AmigaVision moves to this icon, the program ends and the user is returned to the AmigaVision Screen.

Moving downward, double-click on the second **If-Else** icon. You see the Expression Editor with the expression **response() = = "DONE"** in its display area. If the expression is true, (i.e., the user has chosen "Done" on the keyboard), AmigaVision will proceed to the right. If the expression is false (i.e., the user has chosen any key *other* than "Done"), AmigaVision moves downward. Choose Cancel to exit this requester.

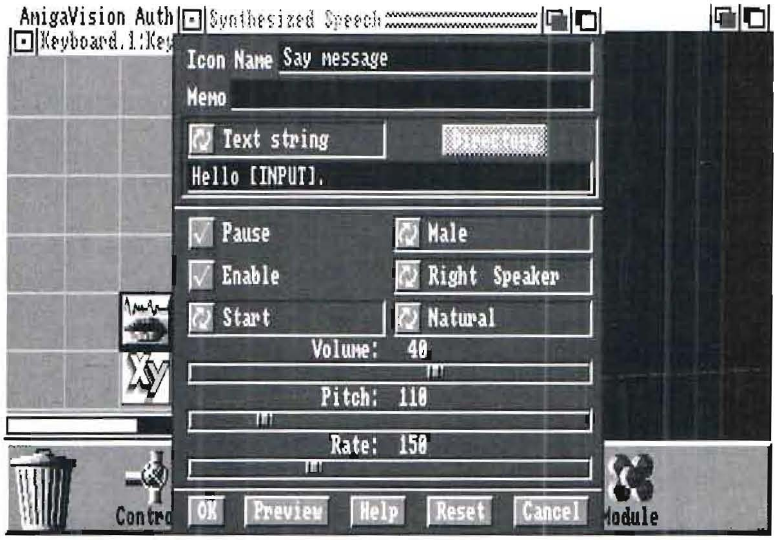
Now look at the Flow Window. If the user selects "Done" on the keyboard, we move to the **End Loop** icon. This ends the current loop (and skips the rest of the children of the second loop—the loop we are presently in). From here, AmigaVision moves to the next child of the *first* Loop icon, which is the **Speak** icon (labeled "Say Message"). Here, the computer speaks the message and loops back to the beginning of the process.

If the user selects another key, AmigaVision moves downward to the next **If-Else** icon. Double click on the icon to see the Expression Editor. The expression **response() = = "BKSPC"** is only true when the user types the backspace key to make this expression true. Select Cancel to exit this requester.

If the user selected the backspace key, AmigaVision would proceed to the **Variable** icon to its right. Double-click on this icon to see the Variables requester. Click on the input string to see the Expression Editor. It has been set to remove the last response from the input string, i.e., the last letter the user entered on the keyboard. These settings come from the Functions list you see on the left side of the Expression Editor. Functions are explained in detail in Chapter 4. Select Cancel to exit the Editor and exit this requester.

If the user had selected a key other than backspace, AmigaVision would have proceeded downward in the Flow Window to the **Variable** icon (labeled "Add next letter"). Double-click on this icon. The expression **input = input + response**() instructs AmigaVision to add the current letter (the RESPONSE variable) to INPUT. In other words, INPUT will equal the concatenation *all* of the letters the user entered in the keyboard, each time AmigaVision loops through that part of the program. Select Cancel to exit the requester.

As you saw, with the second **If-Else** icon and the **End Loop** icon, when the user selects the "Done" key, AmigaVision proceeds to the **Speak** icon (labeled "Say Message"). Double-click on this to enter the Synthesized Speech requester. As you know from previous examples, it is here where you specify what the Amiga will say and set the parameters of the voice. By specifying **Hello [INPUT]** in the text entry area, the Amiga is instructed to say "hello" and then substitute the text string represented by the INPUT variable. Remember, INPUT is the sum of all the responses entered by the user. It is necessary to put square brackets around the the word INPUT to indicate that it is substituted. Select Cancel to exit this requester.



Synthesized Speech

Icon Name Say message

Memo

Text string

Hello [INPUT].

☑ Pause

☑ Enable

Start

Male

Right Speaker

Natural

Volume: 40

Pitch: 110

Rate: 150

The last child of the first Loop icon is the **Variable** icon labeled "Clear INPUT." Double click on this icon to see its requester. The expression **input = " "** clears that variable. It is placed here so that INPUT is cleared and the user can enter a new message in the keyboard.

Once the INPUT variable has been cleared, AmigaVision cycles back to the first **Loop** icon and begins the process again.

Quiz

Remove Tutorial Disk 1 and insert Tutorial Disk 2. Then, load the file QUIZ.AVf in the same way that you loaded the two files on Tutorial Disk 1. Then select **Present . . .** from the **Project** pull-down menu. Make sure to have the volume on the Amiga turned up.

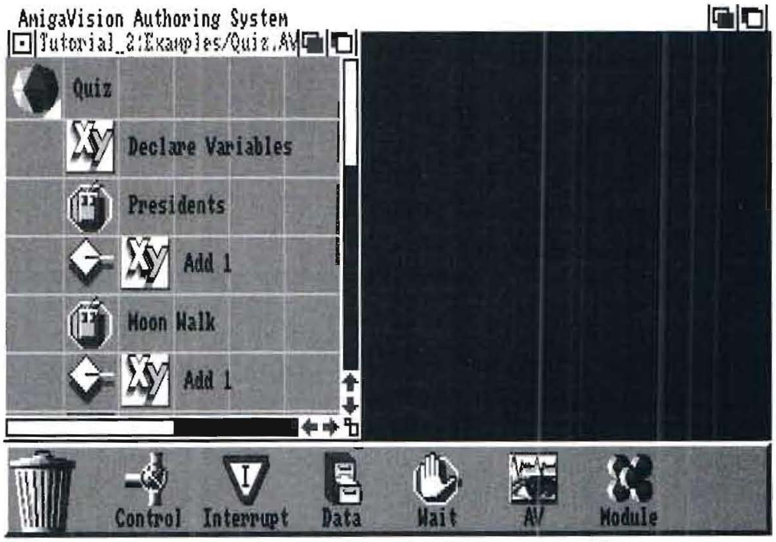
This example is a quiz containing three questions. Don't select the correct answer on each question—experiment and see what happens when you select a wrong answer. When you have answered all three questions, your score will be announced and the program will end.

How Quiz Works

Let's take a look at the Flow Window and see how this application operates. The basic outline of the presentation is as follows:

1. ***Load the first question (called "Presidents") and wait for user to choose an answer.***
2. ***If user picks the correct answer to "Presidents", the sound representing a correct answer is heard and one point is added to his score. If user picks the incorrect answer to "Presidents", the sound representing an incorrect answer is heard.***
3. ***Load the next question (called "Moon Walk") and wait for user to choose an answer.***
4. ***If user picks the correct answer to "Moon Walk", the sound representing a correct answer is heard and one point is added to his score. If user picks the incorrect answer to "Moon Walk", the sound representing an incorrect answer is heard.***

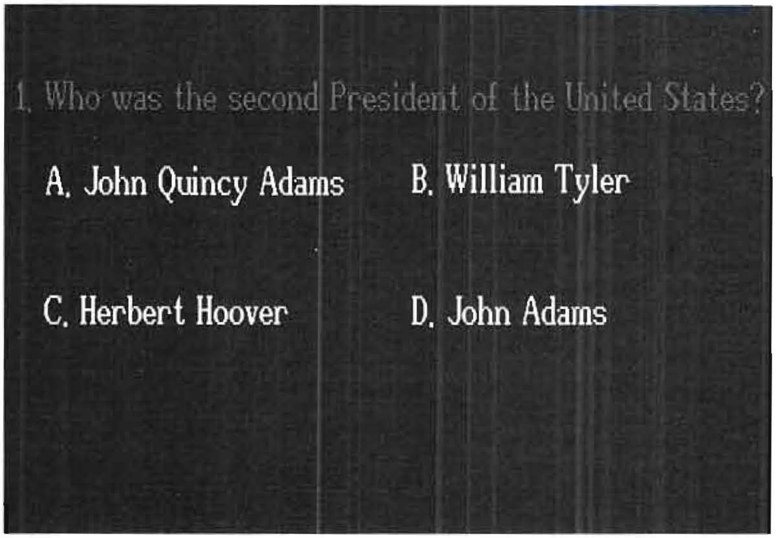
5. ***Load the next question (called "Earth") and wait for user to enter a text answer in the string gadget.***
6. ***If user enters the correct string in the "Earth" question, one point is added to his score. If user enters an incorrect string, no points are added to his score.***
7. ***Load the next screen (called "End Screen") containing the user's score.***



The first icon (after the parent module icon) is a **Variable** icon. Double-click on it to see the Variables requester. This is used in the very beginning of the Flow Window to set the values which will be used throughout the example.

There are two variables in this example—"answer" and "score". The expression **answer = " "** establishes that the variable (in this case, the answer provided by the user is a string (or text) variable. The expression **score = 0** establishes that the user's score is an integer which begins with the value of 0. They must be established as a first step so the program can eventually recognize correct text answers and keep a running tally of the score. Click on Cancel to close the Variables requester.

The next icon down is the **Wait Mouse** icon for the question called "Presidents". Double-click on this icon to enter the Wait Mouse requester, and then click on the **Object Editor** gadget. This brings you to the **Object Editor** screen, where the question was entered and hit boxes were set up.



1. Who was the second President of the United States?

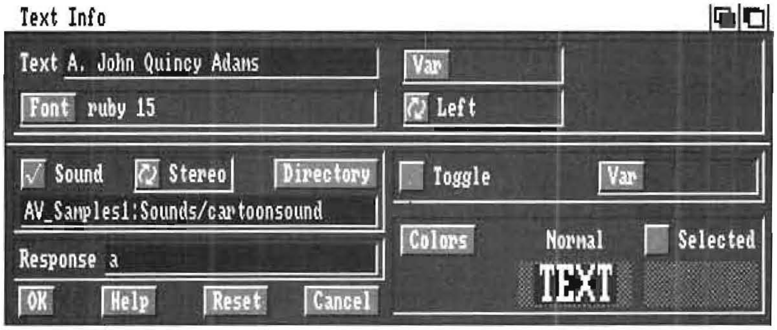
A. John Quincy Adams

B. William Tyler

C. Herbert Hoover

D. John Adams

Double click on Answer A to see its **Text Info** requester. Notice that the gadget next to **Sound** contains a checkmark. This indicates that a sound should be produced whenever this hit box is selected. To its right is a gadget indicating that the sound produced should be in stereo.



Text Info

Text A. John Quincy Adams

Font ruby 15

Left

☑ Sound

AV_Samples1:Sounds/cartoonsound

Response a

The text entry gadget under **Sound** is the space where the name of the sound file is entered. The sound selected for this hit box is called "cartoonsound". Select the **Directory** gadget above the text entry gadget. You may choose your sound file from this **Directory** requester. Exit this requester by selecting the Cancel gadget.

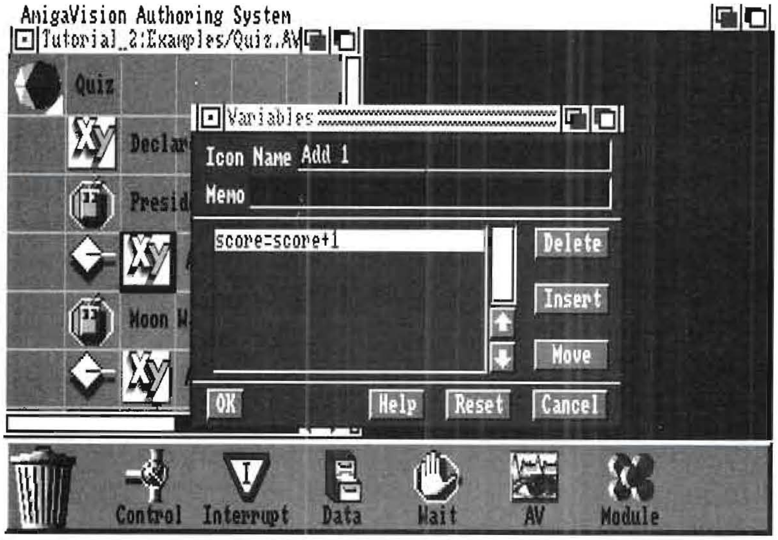
Under the sound gadgets is a text entry gadget for **Response**. In it, you will find the letter that corresponds with the answer you selected. This value is used later to check if the correct answer was given by the user.

Exit this requester by selecting Cancel, and choose Exit from the Project pull-down menu. Also exit the Wait Mouse requester to return to the outline.

Directly under the **Wait Mouse** icon in the flow window, you will find an **If-THEN** icon. Double click the icon to bring up the Expression Editor. It contains the response that AmigaVision is looking for (i.e., the correct answer to the question), evaluated in the expression **response()= = "d"** (this is found in the text box towards the top of the expression editor window).

If the user selected "d" (making the expression True) AmigaVision will proceed with the icon next to the **If-THEN** icon. If the expression is not true (that is, if "d" was not selected) then the expression is false, and AmigaVision continues with the next sibling (the next icon directly below) of the **If-Then** icon. Click on Cancel to close the Expression Editor requester.

Let's say the question was answered correctly. We would then move right to the **Variables** icon. Double-click on this icon to see its requester.



AmigaVision Authoring System

Tutorial_2:Examples/Quiz.AV

Variables

Icon Name Add 1

Memo

score=score+1

The expression **score = score + 1** in its scroll area adds one point to the user's score. (Remember, the first variable requester set up the score variable as **score = 0**.) Exit the Variable Requester by selecting the Cancel gadget.

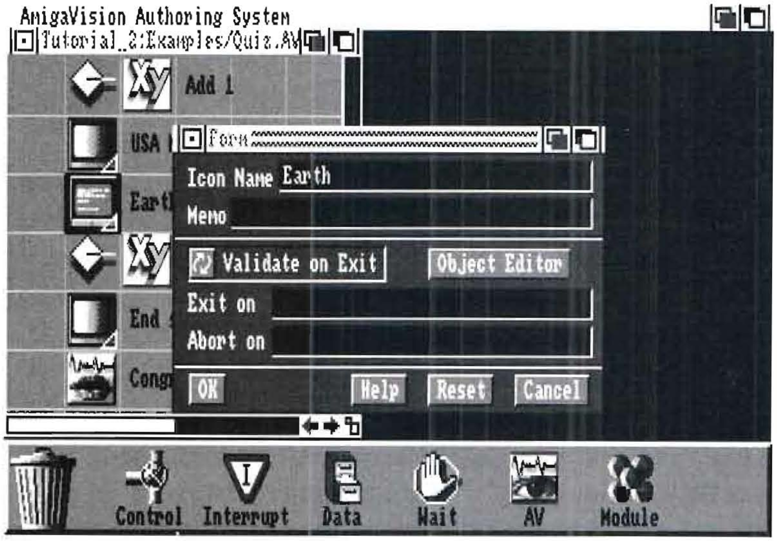
After the question was answered correctly and a point was added to the user's score, AmigaVision proceeds to the next question called "Moon Walk". A user who answered a question incorrectly, (and thus moved downward from the **If-Then** Icon) would have proceeded to "Moon Walk" directly, without having a point added to his/her score.

The **Wait Mouse** icon for the "Moon Walk" question is set up in the same way as the "Presidents" question. Double-click on the **Wait Mouse** icon under the **If-Then** icon.

Now click on the Object Editor gadget. You see the screen which was created for the question. Select a hit box and then select Info from the Objects menu. Notice the various settings for this screen. Select Exit from the Project menu and Cancel to exit from the Wait Mouse requester.

The sibling and partner icons are placed in the same fashion as they are for the "Presidents" question. If the question is answered correctly, we move right in the Flow Window, one point is added to the user's score and we go to the next question (called "Map"). If the question is answered incorrectly, we proceed directly to the next question.

The **Screen** icon labeled "USA Map" loads the background picture for the question. The icon below it labeled "Earth" is a **Data Form** icon. **Data Form** icons are found in the **Data** submenu and are used to create hit boxes in which the user must *type* in information. Double click on the **Data Form** "Earth" icon to see the Form Requester.



AmigaVision Authoring System

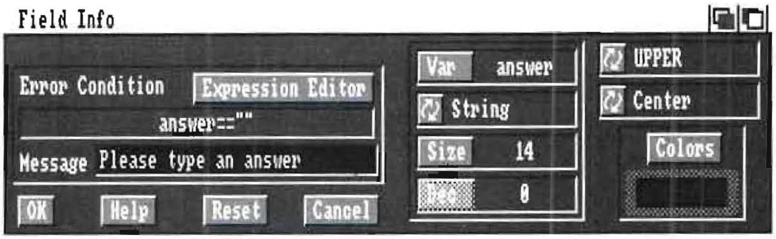
Tutorial_2:Examples/Quiz.AV

Icon Name Earth

After the user enters text and presses Enter, the program is exited.

Click on the **Object Editor** to see the "Earth" screen. Click once on the hit box at the lower left of the screen where the answer is entered and notice how the moving lines run *through* the box. This indicates that the hit box has been set up to receive text.

Now double click on that hit box to see its **Field Info** requester. Click on the **Expression Editor** gadget. The expression **answer= =**" " was set, telling AmigaVision to return an error message if the user presses Enter without entering an answer. Now select the **Cancel** gadget to go back to the **Field Info** requester. The error message AmigaVision will return is entered in the text entry gadget labeled **Message**. This message will tell users to "Please type an answer" if they have not entered an answer and pressed Enter.



Field Info

Error Condition answer==""

Message Please type an answer

Var answer

String

Size 14

Dec 0

UPPER

Center

Exit the **Field Info** requester by selecting the **Cancel** gadget. Choose **Exit** from the **Project** pull-down menu to leave that screen. Click on Cancel to exit the **Form** requester.

Once again, the subsequent icons are set up in the same manner. If the question is answered correctly, we move to the right in the Flow Window and one point is added to the user's score. This time instead of going to another question, we go to the **Screen** icon (labeled "End Screen") which displays the closing screen of the program.

Directly under the "End Screen" icon is a **Speak** icon (labeled "Congrats"). Double-click on this icon. As you learned in the previous example, the **Speak** icon allows you to specify the text that the Amiga will speak, the gender of the speaker, the volume, pitch, and so on. In this example, the Amiga has been instructed to tell the user his final score. By entering the statement "Congratulations, you got [score] right" in the text gadget, the Amiga speaks those words and substitutes the integer value for the variable "score".

Before you go on to Keyboard, click on the Close Window gadget for Quiz.AVf.

States

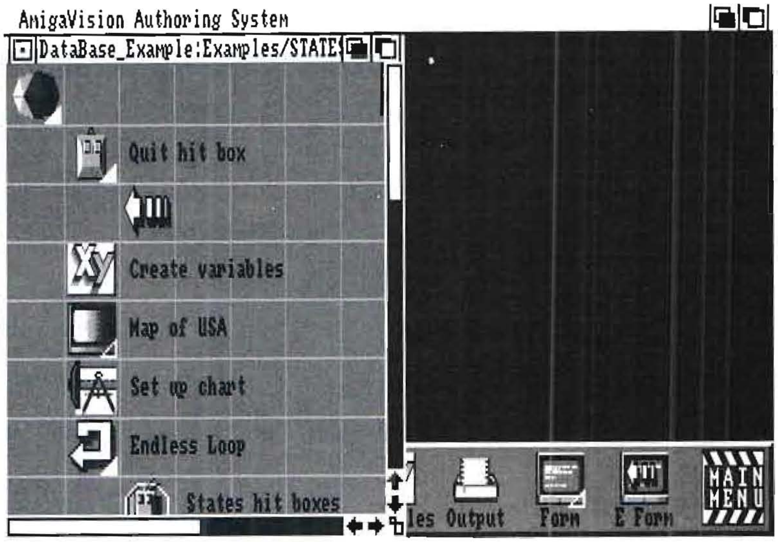
Load the file States.AVf in the same way that you loaded the other files in this tutorial. Then select **Present** . . . from the **Project** menu. Make sure to have the volume turned up.

You will be presented with a map of the continental United States. On this screen, click on a state. The state name and state capital name appear at the top of the screen. Click on states for as long as you like. When finished, click on the word Quit at the lower right of the screen.

How the States Example Works

The heart of this example is a database that contains information about state names and state capitals. You will see how useful a database can be in your presentations. The basic outline of the presentation is as follows:

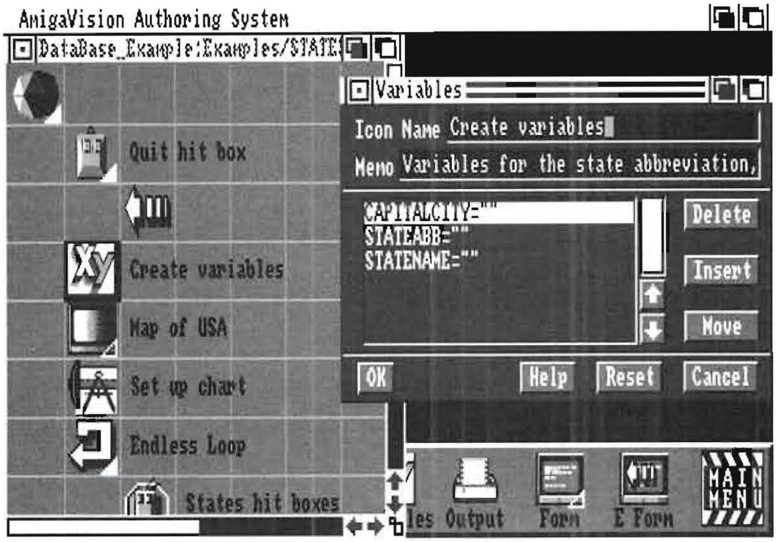
1. ***Set up a Quit hit box which will exit when clicked on.***
2. ***Set up the variables to be used in the presentation.***
3. ***Set up USA map and display objects.***
4. ***Wait for user to click on a state.***
5. ***Assign a state abbreviation to the area clicked by user.***
6. ***Store the hit box response in a variable.***
7. ***Locate the appropriate database record for the proper state, based on the variable just stored.***
8. ***Read the other information contained in the database (state name and capital).***
9. ***Display that information on the screen.***
10. ***Loop back, and give the user the opportunity to click another state.***



The first icon in this flow is a **Mouse Interrupt** icon, which is used to set up an area in the Object Editor that says **Quit** in the lower right hand corner of the screen. Details on using the Object Editor appear in Chapter 4.

This icon will wait for a user to click on that specified region of the screen. When clicked on, the region returns the response "Quit", and the child of the **Mouse Interrupt** icon is performed. This child is a **Quit** icon, which simply exits the presentation.

Next, you encounter a **Variables** icon. Double-click on this icon to display the Variables requester.



Icon Name Create variables

Memo Variables for the state abbreviation,

```
CAPITALCITY=""
STATEABB=""
STATENAME=""
```

Here you define the variable you will be using for your presentation. In this case we have three: CAPITALCITY, STATEABB, STATENAME. At this point in the outline, all three variables have null string values (double quotes with nothing in them). But AmigaVision knows that they will be string variables (that is, variables made up of alphanumeric characters).

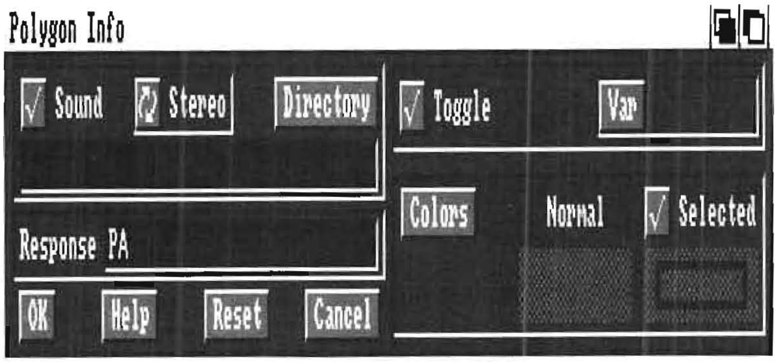
Now we display the USA map with a **Screen** icon. Double-click on the **Screen** icon and notice how the filename has been defined in the same manner as all the other screens in previous examples.

After the background screen has been presented, we want to place a little chart that will display the headings STATE and CAPITAL. This chart was created by placing a **Graphics** icon under the **Screen** icon. The **Graphics** icon was then double-clicked, which opened the Graphics Requester. From here, the *Object Editor* gadget was clicked on to enter the Object Editor, where you can add lines, text, polygons, etc. Enter the Object Editor now in the manner described above, and notice how the chart was placed on the background screen. Select Exit from the Project pull-down menu to go back to the main outline.

Now comes the heart of our presentation. We have set up an Endless loop with a **Loop** icon that has a number of children attached to it.

The first child is a **Wait Mouse** icon. This was used to enter the Object Editor again. This time, the Object Editor was used to Add Polygons. Polygons were drawn around the outlines of all the states borders that appeared on the background screen.

Try double-clicking on the **Wait Mouse** icon, then clicking on the Object Editor gadget. You will see a screen appear with the dotted outlines of the 48 states. These are the polygon objects described above. Now double click on any state.



Polygon Info

☑ Sound

☑ Toggle

☑ Selected

Response PA

A Polygon info requester will open on the screen. Note carefully the settings in the requester, especially in the response box. For example, if you clicked on the Pennsylvania area, you will find the letters PA in the response box. This becomes significant later.

Exit from the Object Editor and return to the main outline (Select Cancel, then Exit from the Project pull-down menu).

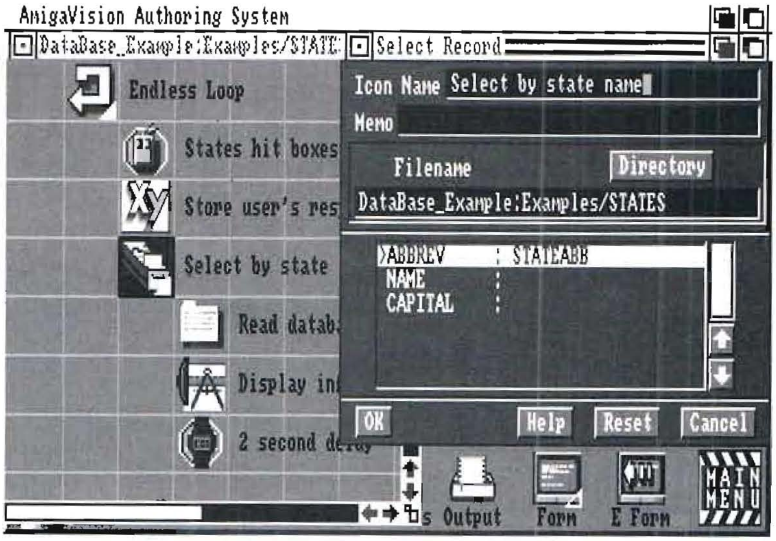
Below the **Wait Mouse** icon (which has now set up all the states as hit boxes) is another **Variables** icon. This particular **Variables** icon is used to store the response in one of the variables previously defined.

Double-click on the **Variables** icon. In the requester you will find that the following expression has been assigned.

STATEABB = response()

Here you are assigning the two letter state abbreviation of whatever state has just been clicked on to the variable STATEABB. For instance, if Pennsylvania had been clicked on, it would have returned the response "PA", and AmigaVision would then make STATEABB = PA.

Now we can use our database to find other information about Pennsylvania. The next icon down is called a **Select Record** icon. This is used in conjunction with databases. Double-click on this icon. A requester will open showing you the name of the database file being used, and the three fields that make up this database.



Icon Name Select by state name

Memo

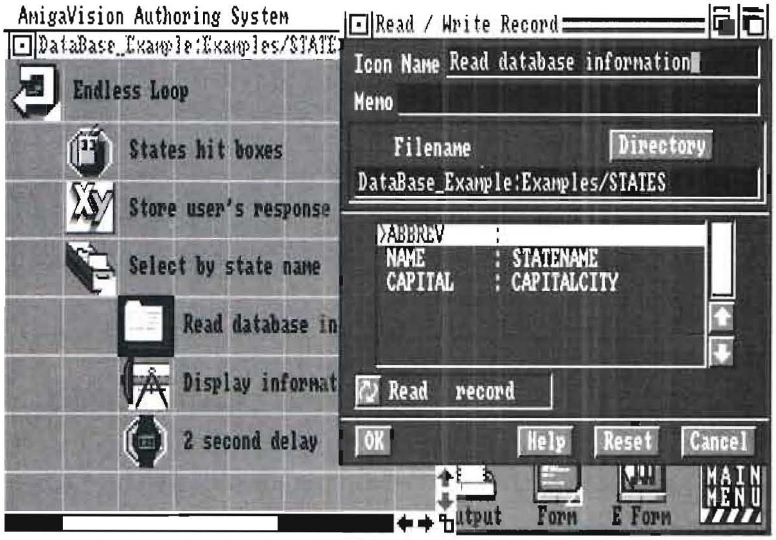
Filename DataBase_Example:Examples/STATES

ABBREV	: STATEABB
NAME	:
CAPITAL	:

Remember that a database has previously been created which contains information about the United States. In your applications you would first create the database, then build applications that used the database. You can use AmigaVision itself to create your own databases (see Chapter 4).

Note how in this requester the ABBREV field in database has been specified to use the STATEABB variable. What this means is that AmigaVision will select or search for the database record by looking at the ABBREV field (state abbreviations) and matching it to the STATEABB variable assigned above. In our example, the STATEABB variable has the value "PA". The **Select Record** icon searches the database for ABBREV fields that match the variable "PA". Of course, only one record will match, which is precisely what we want.

Once the record has been selected, various things can be done to it. In this instance, a **Read/Write** icon is used to read additional information from the record into AmigaVision. Double-click on the **Read/Write** icon. A requester will open. Notice how now the two other fields in the database have been assigned corresponding variables.


AmigaVision Authoring System

DataBase_Example:Examples/STATE

Endless Loop

States hit boxes

Store user's response

Select by state name

Read database in

Display informat

2 second delay

Read / Write Record

Icon Name Read database information

Memo

Filename

DataBase_Example:Examples/STATES

```
ABBREV   :
NAME     : STATENAME
CAPITAL  : CAPITALCITY
```

Read record

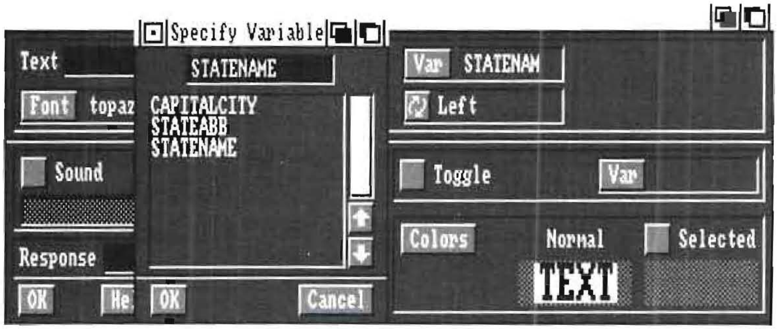
This icon will simply read the NAME field into the variable STATENAME, and also read the CAPITAL field into the variable CAPITALCITY. Once again, to use our Pennsylvania example, the **Read/Write** icon reads the NAME field ("Pennsylvania") into STATENAME, and the CAPITAL field ("Harrisburg") into CAPITALCITY.

Note that all three variables that were originally setup at the beginning now have values.

Now that we have the information we want, we need to display that information. The next icon, another **Graphics** icon, uses the Object Editor to place the values for the STATENAME and CAPITALCITY variable into the correct area on the screen.

Double-click on the **Graphics** icon, then in the new requester, click on the *Object Editor* gadget to enter the Object Editor.

Notice how the variable names themselves have been added to the screen. Double-click on STATENAME. A Text Info requester opens.



In the upper right corner of this requester click on the *Var* gadget. Notice how the text has been specified to use the STATENAME variable. What this means is that AmigaVision will substitute the value of the variable in this text area. So in our example, "Pennsylvania" will appear where we now see the words STATENAME. The CAPITALCITY field works in the same fashion.

The last icon in our loop is a **Delay** icon, which is simply used to add a two second delay to the display before AmigaVision loops back to the beginning. This will allow you time to read the Statename and Capital city.

When the loop begins again, the States are re-displayed, and the user can click on another state. The loop will continue until Quit is selected.

You may want to run the presentation again, and study it some more. Some other interesting features you might want to explore are the use of feedback colors and sounds. This happens when you click on a hit box during a presentation, and can be controlled from the Info Requesters for hit boxes in the Object Editor.

Another way to select sounds is to use the Defaults option from the Project pull-down menu. Here you can specify a path and filename for a sound. This way you don't have to specify the same sound over and over again if you don't want to.

Please also refer to Chapter 4 and Chapter 6 for more information about using databases with AmigaVision.

Chapter 3

How AmigaVision Works

Chapter 3: How AmigaVision Works

With multimedia authoring, you need to plan your presentation much like a flow chart where one action flows logically into another. Once you have a stated objective for a multimedia application, you use AmigaVision to outline the flow of information. AmigaVision provides a **flow** (or outline) **editor** for this purpose.

You can also collect the various elements you will be using for your presentation (i.e., pictures, sounds, music, video, text). For this purpose use AmigaVision's **content editor** to assemble and organize audio-visual elements that were created using other graphics, animation, audio and text editing software.

You organize the outline and the content information in AmigaVision using a set of icons. Some of these icons represent the actual *content* of the presentation, such as video, graphics, animation, sound, and text events and commands. Other icons represent the *flow* and *structure* of the presentation and are used to build control structures.

Once you have your flow and content ready, you may immediately run the presentation to see if it needs refinement. Thus the development process becomes highly interactive, and you may validate your work in small segments, minimizing errors in logic and flow.

When you are ready to end an AmigaVision session, you simply save your work. When you are satisfied that the application is ready for use, use AmigaVision's **Runtime** feature to prepare the presentation or course for actual use and distribution.

Here's a scenario for how you might design an interactive lesson:

1. ***Create an outline for the presentation, and break it down into sub-outlines.***

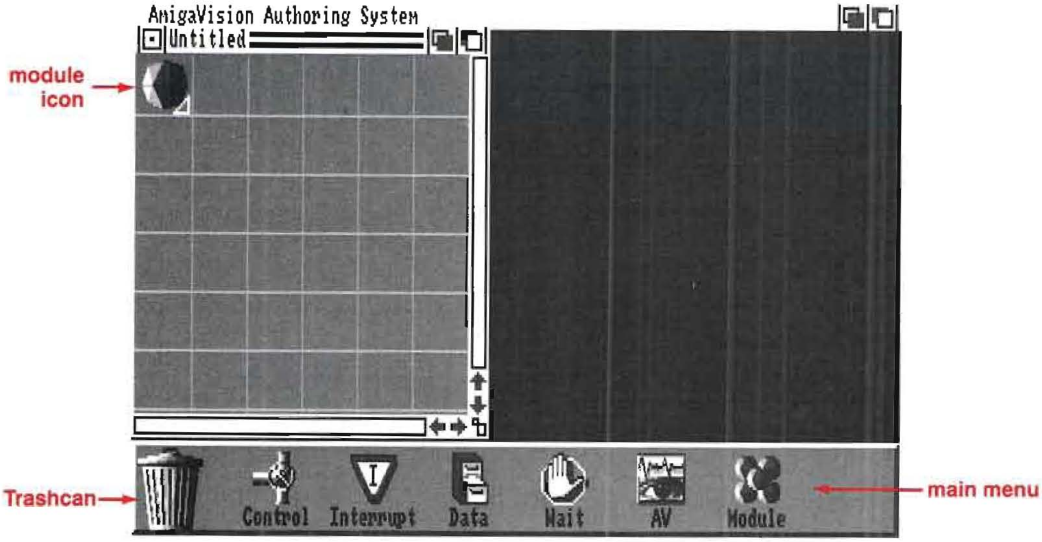
2. ***Assemble the contents of the presentation (which have been created in other packages like word processors, paint programs, etc.) in the content editor.***

3. ***Outline the actual presentation in the flow editor.***
4. ***Run and refine the presentation.***
5. ***Use the Runtime feature to prepare the presentation for actual use.***

Using AmigaVision

When you start AmigaVision, the first screen you see contains a Flow Window labeled "Untitled." This window contains a **Module** icon. A module icon represents a major section or subsection of your presentation.

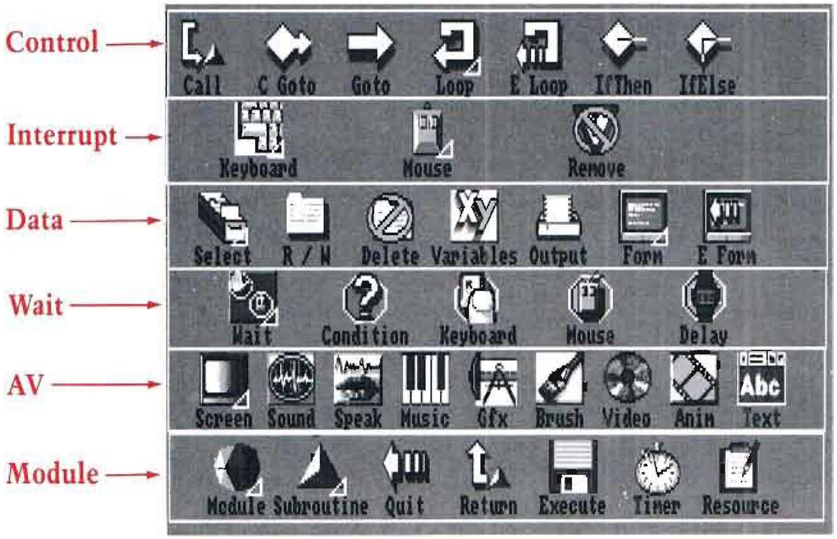
The **Trashcan** icon appears at the lower left corner of every screen. Next to the Trashcan at the bottom of the screen is the Main Menu of icons. On the AmigaVision Main Menu there are six other icons, each of which represents a submenu of icons.



Manipulating Icons

In order to get around in AmigaVision, you need to familiarize yourself with its icons. Remember that the icons at the bottom of the opening AmigaVision screen represent six different submenus.

AmigaVision Submenu Icons

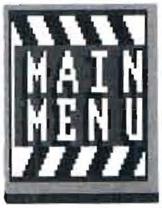


To see a submenu of icons, click once on one of the Main Menu icons. Icons are organized into groups by major functions, such as Control and AV, to allow you to easily find the appropriate icon to include in your outline. Each submenu contains its own row of icons.

You create the outline of your project by dragging icons from the submenus and placing them in the Flow Window. To drag an icon, move your mouse pointer to the icon, click on the icon with the left mouse button, and while holding down that button, move the icon into the flow window.

When you place an icon in the outline, that icon is automatically selected. The selected icon is indicated by a dark rectangle in the icon background. To select a different icon, click on it. You may select only one icon at a time.

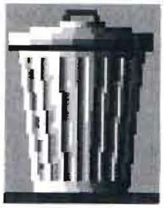
Main Menu



In each submenu you will see a **Main Menu** icon on the far right. From each submenu you have the option to go back to the Main Menu by clicking the **Main Menu** icon. For example, to go from the AV submenu to the Control submenu:

1. ***Click on the Main Menu icon, which takes you back to the Main Menu.***
2. ***Click on the Control icon, which takes you into the Control submenu.***

Trashcan



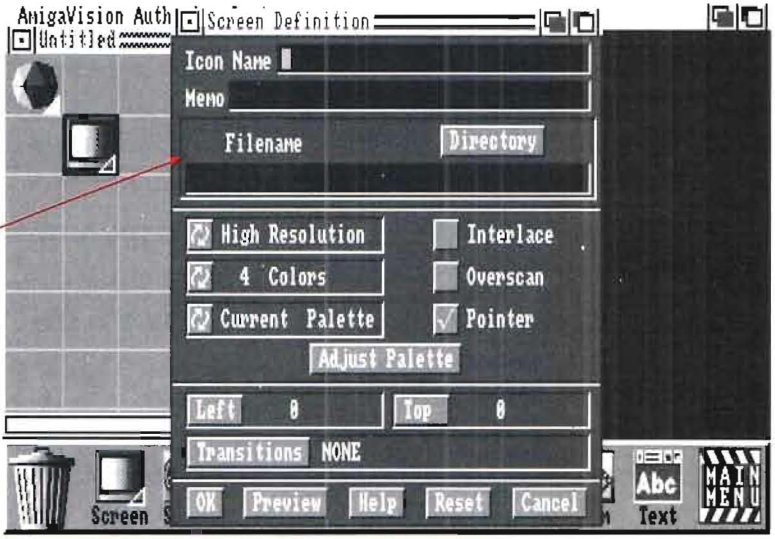
Any time you wish to remove an icon from your flow outline, you simply drag it to the **Trashcan** icon. Any icons thrown into the trashcan cannot be recovered.

Defining an Icon

Imagine that you are at the Main Menu, and you want to display a screen in your presentation. First, you click on the **AV** icon to enter the AV submenu. Then you drag the **Screen** icon from the menu and place it in the Flow Window diagonally and down to the right from the **Module** icon. The **Screen** icon drops into place at the lower right of the **Module** icon.

Here's what you have done to this point. You have set up the presentation to display a picture, but AmigaVision does not yet know which picture to display. Therefore you must **define** the icon, which means that you specify exactly what action is going to be performed by the icon (in this case displaying a particular picture file).

To define an icon in your outline, double-click on the icon. A Definition Requester will be displayed on the AmigaVision screen. Here you define which file you wish to display or the specific action you want the icon to perform.



Icon Name

Memo

Filename

High Resolution

4 Colors

Current Palette

□ Interlace

□ Overscan

☑ Pointer

Left 0

Top 0

Transitions NONE

definition requester for screen icon

Icon Relations

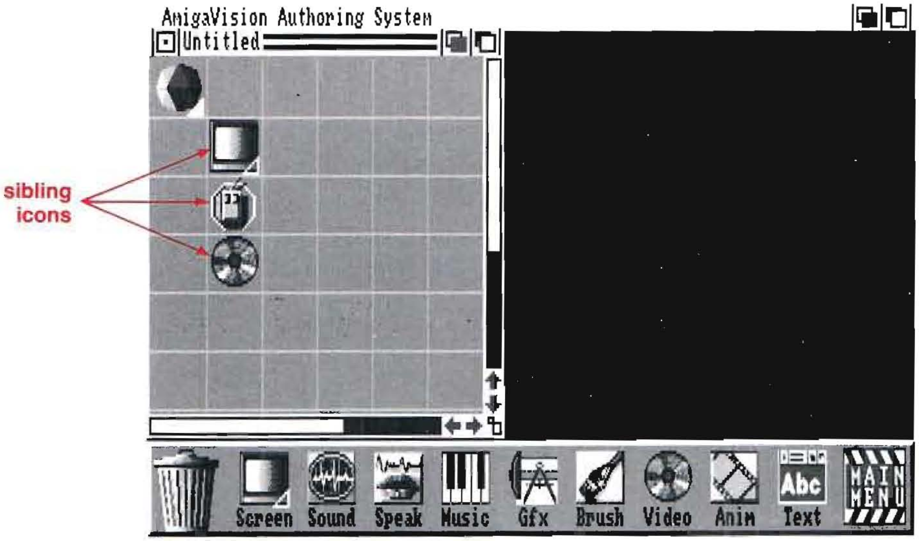
Icons are placed in the Flow Window in relation to other icons. These relationships are akin to the relationships in a family tree. There are four types of icon relations in AmigaVision: sibling, parent, child, and partner.

Icon relations and placement are illustrated in the following examples.

Sibling Icons

Icons placed one below the other in the outline are referred to as **siblings**. To place an icon as a sibling, drag and place the icon inside the square directly above or below another in the outline. As long as the tip of the pointer is within the target square, the icon will align itself properly.

If you drop an icon into a square already occupied by another, the new icon will drop into the target square and "push down" the old icon and all icons below it in the flow.

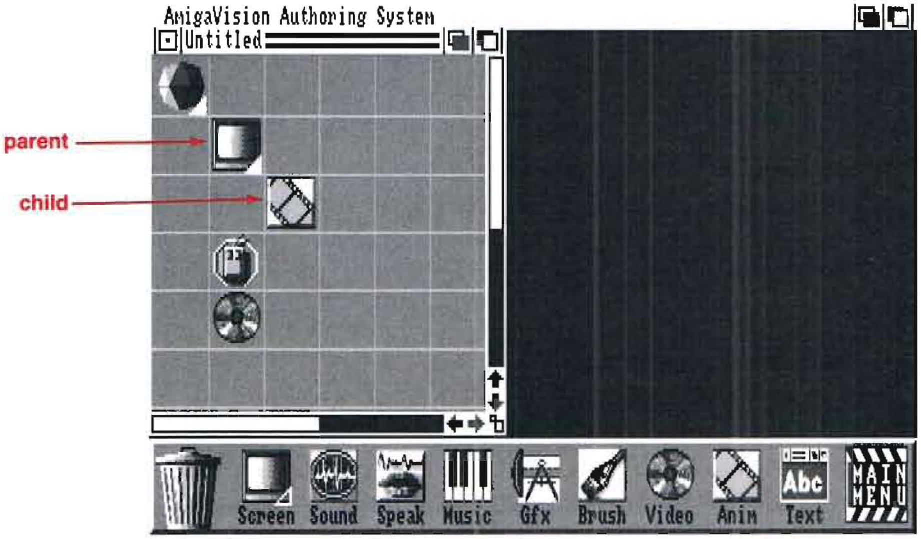


During presentation, the sibling icons are executed from top to bottom in the order in which they are encountered.

Parent And Child Icons

Certain icons may be parents to other icons in the outline. Potential parent icons have a small triangle in the lower right corner. When the parent has children icons, the triangle becomes solid. Think of the children icons as being contained by or belonging to their parent.

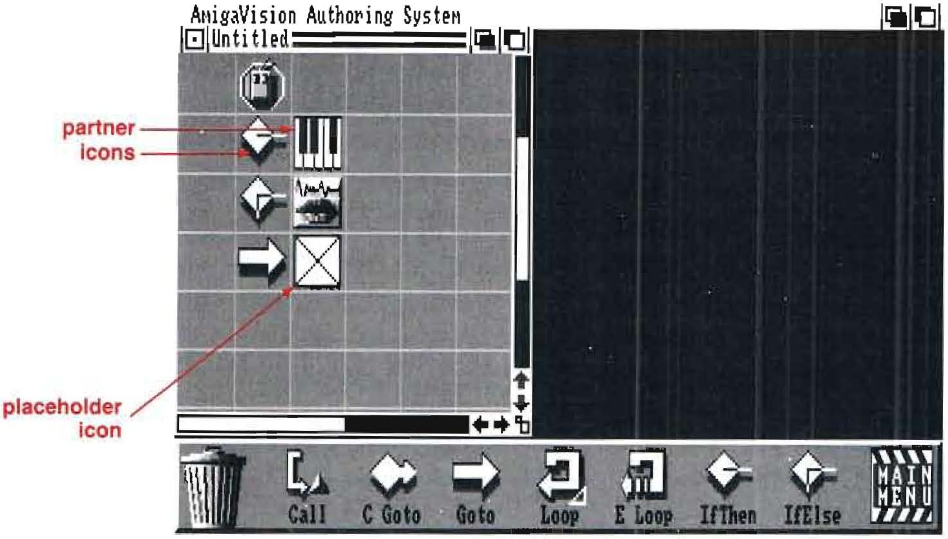
To place an icon as a child, drag and place the icon one square to the right and one square down (that is, directly diagonal) from the icon that will be its parent. As with sibling icons, only the tip of the pointer needs to be within the target square for correct placement.



AmigaVision runs your application by following each layer of the flow outline before advancing in a downward direction. It starts with the topmost parent icon and performs its action, then looks to see if that icon has any children that need to be run. AmigaVision moves down through the outline in this fashion: from Top to Bottom, and Left to Right.

Partner Icons

Certain Control icons (If-Then, If-Else, Goto, CGoto, and Call) require that a partner be placed beside them. To place an icon as a partner, drag it and place it in the square directly to the right of the icon already in the outline.



During the presentation, the left partner runs first, then the right partner. For example, If-Then and If-Else icons evaluate an expression. If true, AmigaVision proceeds to the right partner.

Call, Conditional Goto, Goto icons require a partner which is a reference to an icon elsewhere in the outline. When you place one of these in the outline, a **Placeholder** icon appears as its partner. The **Placeholder** icon remains until you have selected the reference icon.

An icon may have only one partner, and the right partner may not have siblings. If you need to have the right partner perform more than one action, use a **Module** icon and create children of that Module icon to perform each of the actions.

Scrolling the Edit Window

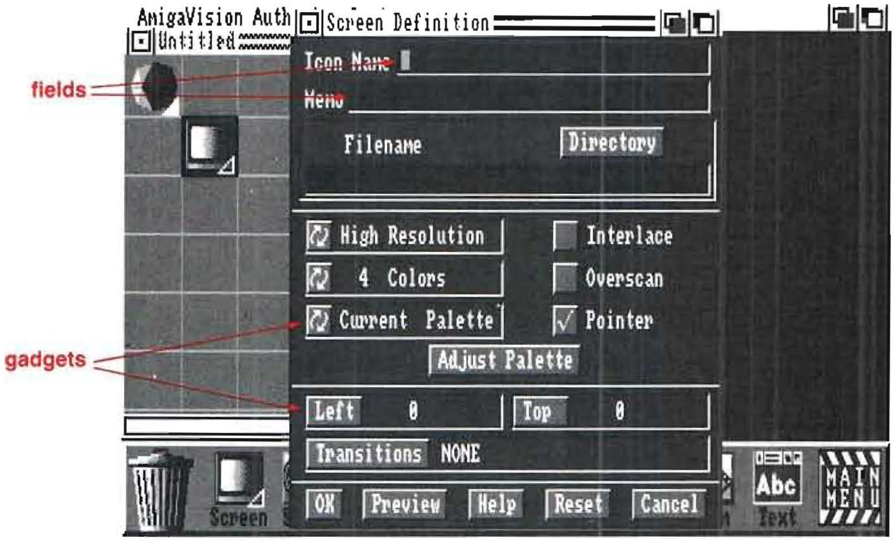
When your application becomes larger than the area displayed by the Edit Window, you can scroll the window in the following manner to move or copy icons:

1. ***With a click of the left mouse button, select the icon you want to move or copy.***
2. ***Drag the icon outside of the window border in any direction and, while holding down the left mouse button, press the right mouse button (or the arrow keys on your keyboard) to scroll the flow outline.***

You may also scroll the window using the directional arrows on the keyboard.

Using Requesters

When you double-click on an icon (to define it, for instance), a *requester* will be displayed on the screen. Requesters confirm your choices or solicit additional information. As shown in the following example, requesters contain *fields* into which you type information, and *gadgets* to indicate your choice of action.



fields

gadgets

Screen Definition

Icon Name

Memo

Filename

High Resolution

4 Colors

Current Palette

☐ Interlace

☐ Overscan

☑ Pointer

Left 0

Top 0

Transitions NONE

Click on a black field with the left mouse button to type information into it. When you press the *Enter* key, the cursor will move to the next field. To choose a gadget, click on it with the left mouse button.

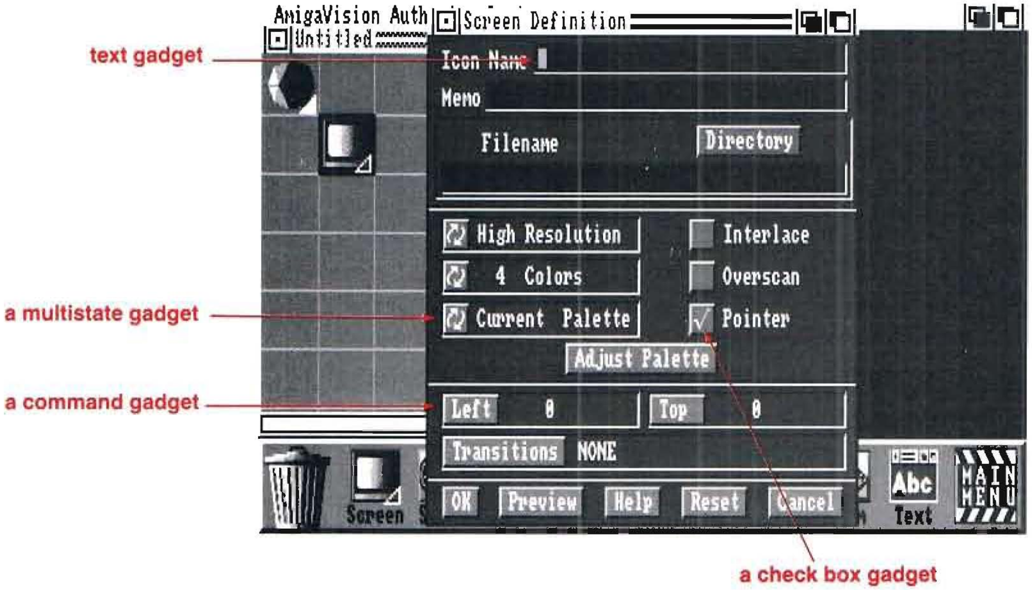
There are three ways to close a requester:

1. ***To save your choices and close the window, click on the OK gadget.***
2. ***To cancel your choices and close the requester, click on the Cancel gadget.***
3. ***You may also cancel your choice and close the requester by clicking on the Close Window gadget at the top left-hand corner of the requester.***

Fields and gadgets common to many requesters are discussed in the following section.

Requester Gadgets

The gadgets in AmigaVision's requesters have been specifically designed to provide a consistent user interface. There are four basic gadget types: **command, multistate, check box,** and **text gadget.** The first three gadgets resemble buttons; the text gadgets are black rectangles.



text gadget

a multistate gadget

a command gadget

a check box gadget

AmigaVision Auth

Untitled

Screen Definition

Icon Name

Memo

Filename [Directory]

High Resolution

4 Colors

Current Palette

□ Interlace

□ Overscan

☑ Pointer

[Adjust Palette]

Left 0

Top 0

Transitions NONE

Command Gadget

Immediately initiates some action when you click on it. A common example is the *Directory* gadget in a requester, which is used to open up the **File Requester** (a new window in which you specify files for your presentations).

Multi-state Gadget

Also called a Cycle Gadget, the Multi-state Gadget consists of cycling arrows on its face, indicating that the selection next to the gadget is one of several possible choices. Each click on the gadget shows you the next choice in the list. For example, in the **Screen Requester** a multistate gadget is used to set the screen resolutions. It cycles from "High Resolution," to "ExtraHalfbrite," to "Hold and Modify," to "Current," to "File Defined" to "Low Resolution."

Check Box Gadget

Toggles a requester setting to an "on" or "off" state. For example, many of the Audio Visual icons have a *Pause* selection. When "on," the on-off gadget shows a checkmark on the top, and when "off," the gadget appears blank.

Text Gadget

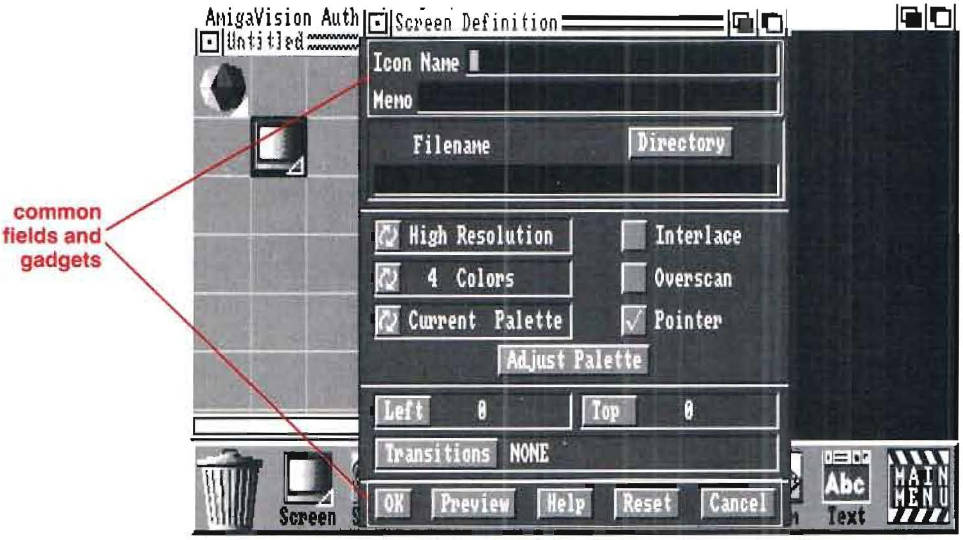
Does not have a button-like appearance. Technically known as a *string gadget,* the text gadget normally appears in requesters as a black area in the middle of an inwardly bevelled rectangle. For example, all icons have an *Icon Name* field which is a text gadget. To change the text in it, click inside the box and type or edit normally. If a genlock is activated, this area is transparent to the video imagery behind it.

Tip: To clear a text field quickly use Right-Amiga X key combination.

In some text gadgets, such as *Memo,* you may type a string of characters longer than will visibly fit in the field. As you type beyond the right edge, the text scrolls. Use the arrow keys to scroll the text in either direction. If there are multiple text gadgets in a requester, hitting the Enter key will advance the cursor to the next one, so you don't need to click in each box with the mouse.

Fields and Gadgets Common to Many Requesters

You will encounter the following fields and gadgets often in AmigaVision. Take some time to acquaint yourself with their functions.



Screen Definition

Icon Name

Memo

Filename Directory

High Resolution

4 Colors

Current Palette

□ Interlace

□ Overscan

☑ Pointer

Adjust Palette

Left 0 Top 0

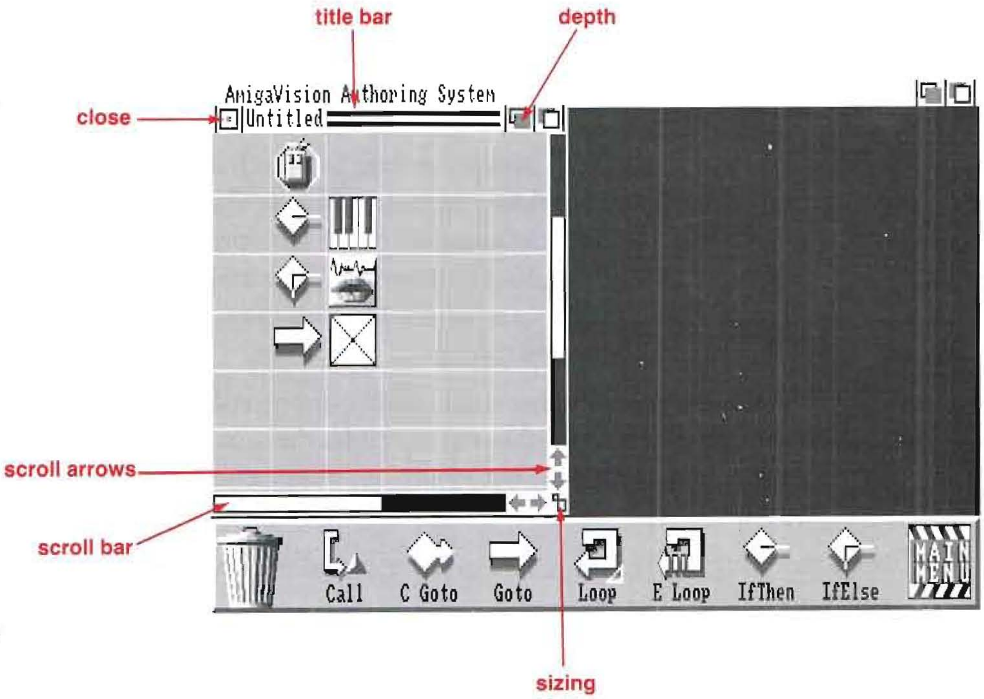
Transitions NONE

OK Preview Help Reset Cancel

- **Icon Name**. An optional name for the icon. The name will appear next to the icon in the Flow Window. It is used to identify and reference the icon and may be up to 31 characters long. This is a very useful feature when you have many similar-looking icons in your Flow Window; you'll be able to distinguish between them if they have names.
- **Memo**. Remarks about the icon for your reference. Remarks are optional and will only appear in the requester for the icon. The memo may include up to 80 characters. The information contained in the *Icon name* and *Memo* fields can be printed out by choosing the **Print** option in the Project pull-down menu.

- **OK**. Saves the current information for the selected icon and exits the requester.
- **Preview**. Performs the action described by the requester and returns to the requester.
- **Help**. Displays a help screen for the current requester.
- **Reset**. Restores the requester to the original state.
- **Cancel**. Cancels modifications, returns to the previously defined options, and exits the requester.

In requesters and windows, there are several gadgets or areas which, when clicked on, perform certain functions. To activate a gadget, point to it and click with the left mouse button.



- **Close gadget**. The Close gadget appears in the upper left corner of a window. Click on it to close the window. Like the Cancel gadget, it will not save any changes you have made.
- **Depth gadget**. The Depth gadget appears in the upper right corner of a window. Click on the Depth gadget to move the window behind and in front of other windows.
- **Title Bar gadget**. The Title Bar gadget appears at the top of the window. It is used to move the window around on the screen. Click on the bar with the left mouse button, drag the window to a new location, and release the button.
- **Sizing gadget**. The Sizing gadget appears in the lower right corner of a window. To size a window, click on the gadget and stretch or shrink the window to the desired size, then release the button.
- **Scroll bars**. The Scroll bars are rectangles which appear along the right and bottom edges of windows. They may be used to scroll over the contents of the window to reveal areas hidden because of the window's current size. Drag the bar up/down or left/right to scroll the window in the direction you desire.
- **Scroll arrows**. Scroll arrows are located at one end of a scroll bar. Clicking on the arrows scrolls the window in the direction indicated, but in smaller increments than with the scroll bars.

Note that gadgets are only active in the selected window. To select a window, click on it.

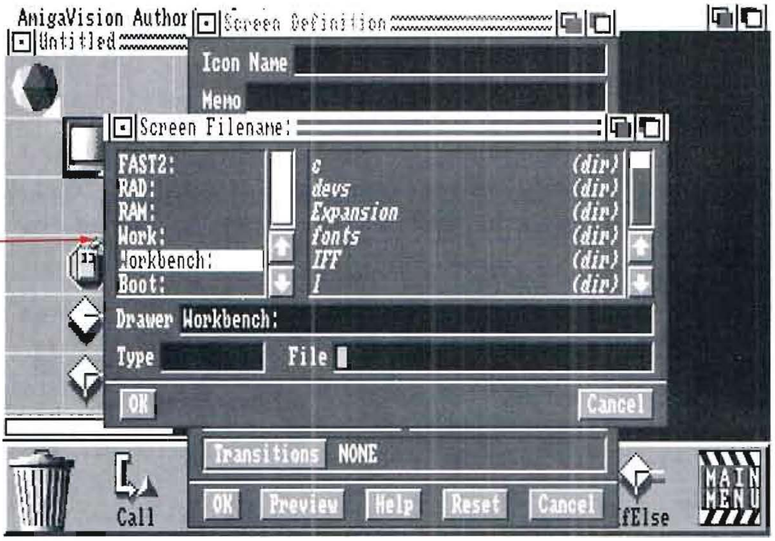
Frequently Used Requesters

You'll encounter the following requesters often when using AmigaVision:

- **File requester**. Used to specify a file in AmigaVision. It opens when you click on a *Directory* gadget in an icon requester.
- **Specify Value requester**. Used when a numerical value is needed. It opens when you click on a gadget that requires a value.
- **Referencing requester**. Used to reference an icon which appears elsewhere in the outline. The first referencing requester opens when you click on the **Placeholder** icon. The second Referencing requester opens when you have selected the icon that you wish to reference.
- **Transitions requester**. Used to define the transitional screen between pictures. It opens when you click on a Transitions gadget in an icon requester.
- **Help requester**. Used to provide context-sensitive information whenever the Help gadget is clicked in a requester.

The File Requester

To specify a filename, click on a *Directory* gadget in an icon's requester. A **File requester** will open.



A File requester

Drawer Workbench:

Transitions NONE

The list on the left contains the names of the volumes (drives or partitions) available on the system. When you click on the desired volume, its name will appear in the Drawer field. The names of the files and directories in the selected volume will appear in the right window. Directories contain files and are indicated by *(dir)*. Use the scroll bar to view all of the files. Click on a directory to view the files in it. To select the file you want, click on its name. The name will appear in the File field.

Use the Type field to specify the kind of files to be displayed. Click on the field, then type in the desired file extension. After this, only files with this extension will be displayed in the right window. A period denotes the beginning of an extension. An example is .AVf, the extension for AmigaVision flow files.

To select a file and exit the requester, either double-click on the filename, click once on the filename then click on the OK gadget or type in the filename and press ENTER. To cancel the selection and exit the requester, click on Cancel.

Variables in Filenames

When specifying a filename for an Audio Visual or Database icon you have the option to use *variables* in the name. This allows a presentation to adapt "on the fly" to user choices.

For example, you create a number of hit boxes that are presented to the user, and a picture will be shown based on the user's selection. You could specify multiple **If-Then-Else** icons leading to different **Screen** icons for each picture, or you could create a single **Screen** icon with a variable filename and have each hit box set the variable to a different filename.

Earlier in the flow you would have to declare a string variable by using a **Variables Icon**. Double-click on this icon to open its requester. Select **Insert**, type pic=" ", select **OK**, then select **OK** again to exit the requester.

To type a variable in the *Filename* field, surround the variable names with brackets. If the variable name is "pic", type [pic]. In our example we would have each hit box define "pic" to equal a different string and the **Screen** icon would substitute that string for the filename.

You may also mix variables with other characters in the filename. In our example all the picture files may be stored on the Work: hard disk in the *Pics* directory. Our filename field would then read:

```
Work:Pics/[pic]
```

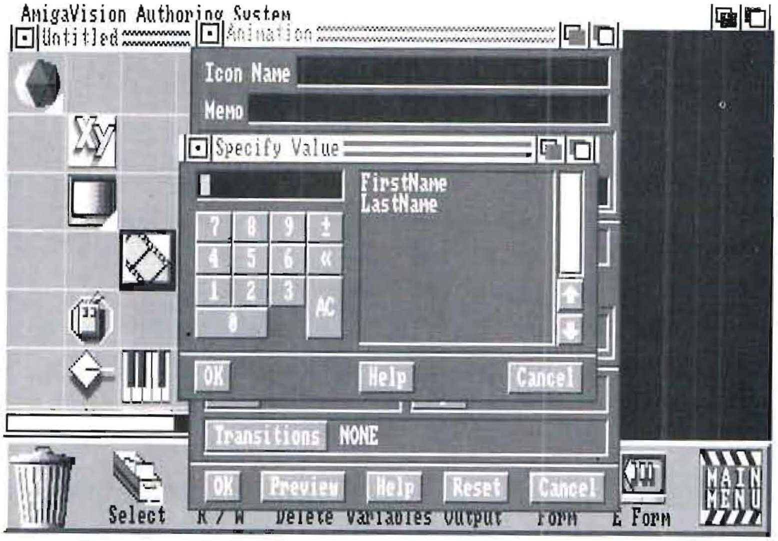
You may mix as many variables as you like in the *Filename* field, and each will be substituted with the current value when you select **Present** . . . .

Note: For AmigaVision to properly display a variable named Picture, you must click on the resolution gadget and select "File Defined."

Specify Value Requester

Suppose you need to enter a video frame number or the number of times an animation will loop during your presentation. When you click on a field requiring a numeric value, you'll open the **Specify Value** requester. The value may be a number or a variable and may be entered in any of the following ways:

- ***Type the number or variable name in the text gadget at the top of the requester.***
- ***Click on the number gadgets to enter the value into the field.***
- ***Click on a variable name from the list at the right. The list contains all of the variables defined for the current position in the outline. Click on a variable to enter it into the field.***



You can exit the requester in the following ways:

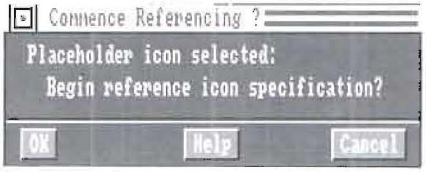
- *To use the new value, click on the OK gadget or press the Enter key on your Amiga keyboard.*
- *To leave without changing the value, click on the Cancel or Close Window gadgets.*

Referencing Requesters and Placeholder Icons

The **Goto**, **Conditional Goto,** and **Call** icons require a partner, which refers to another point in the outline. For example, the **Goto** icon needs to tell AmigaVision where it wants to go. When you place one of the above icons in an outline, a **Placeholder** icon appears.

1. ***To begin the referencing process, double-click the left mouse button on the Placeholder icon.*** This opens a requester which asks "Begin reference icon specification?"

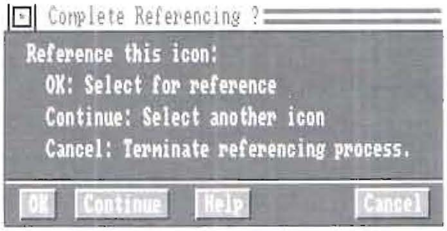
   - ***Click OK to begin the referencing process, or***
   - ***Click Cancel to abort the selection process.***



Placeholder icon selected:
Begin reference icon specification?

2. ***Move around in the Flow Window to find the icon you want to reference and double-click on it.*** A second requester will open. You will be prompted "Reference this icon."

   - ***Click OK to choose the selected icon as the reference. The reference icon will then replace the Placeholder.***
   - ***Click Continue to search for a different reference icon.***
   - ***Click Cancel to end the referencing process.***



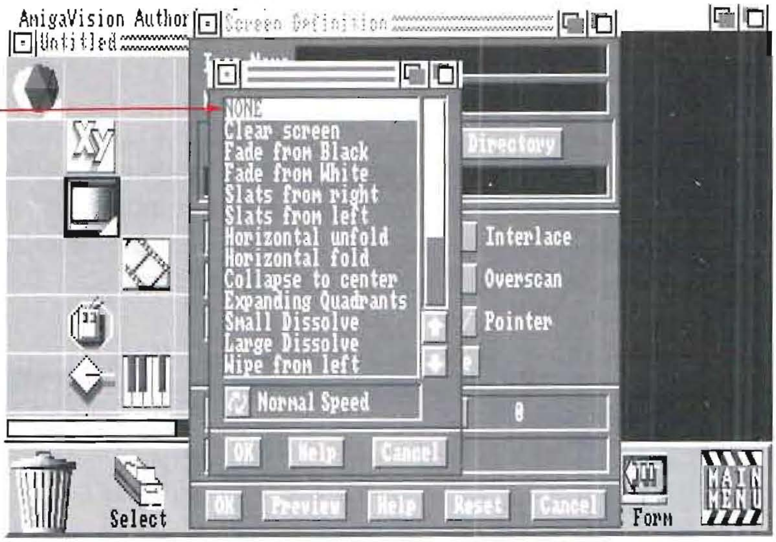
Reference this icon:
OK: Select for reference
Continue: Select another icon
Cancel: Terminate referencing process.

After you have completed this process, the reference icon will appear beside its partner, as well as in its original location in the outline.

Transitions Requester

The Transitions requester allows you to select one of several transitional screen patterns. A transition pattern is what the user sees when the screen changes from one picture or animation sequence to another picture or animation. The available transition patterns are shown below.

transitions requester



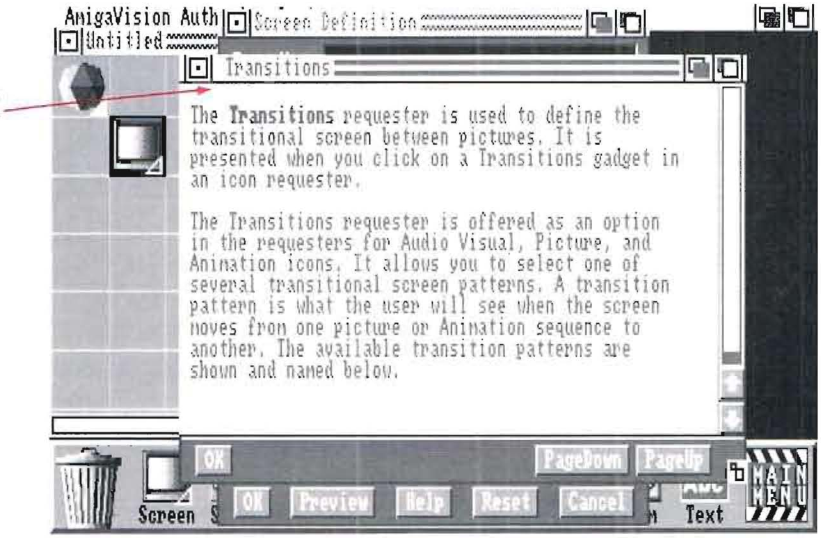
NONE
Clear screen
Fade from Black
Fade from White
Slats from right
Slats from left
Horizontal unfold
Horizontal fold
Collapse to center
Expanding Quadrants
Small Dissolve
Large Dissolve
Wipe from left

Assume, for example, you have placed a **Screen** icon in your outline, and you have defined the file you want that icon to display. Now you want to create a transition. In that icon's requester, click once on the Transition gadget; the Transitions requester will open.

- *Select the transition you want by clicking on its name in the list.*
- *Use the Speed gadget to specify the speed at which the transition occurs — Normal, Fast or Slow.*
- *Click OK. You will return to the Icon requester. The transition you have chosen will appear in the field next to the Transition gadget.*

Help Requester

When you click the Help gadget, AmigaVision opens a Help Requester Window. AmigaVision's help information is thorough and will answer many of your questions without requiring you to seek answers from this manual.



The **Transitions** requester is used to define the transitional screen between pictures. It is presented when you click on a Transitions gadget in an icon requester.

The Transitions requester is offered as an option in the requesters for Audio Visual, Picture, and Animation icons. It allows you to select one of several transitional screen patterns. A transition pattern is what the user will see when the screen moves from one picture or Animation sequence to another. The available transition patterns are shown and named below.

a help requester

You can browse this window with directional arrow gadgets, scroll bars, as well as with the *PageUp* and *PageDown* gadgets. The window may also be resized with the sizing gadget in the lower right corner of the window.

Exit the window by clicking OK or the Close Window gadget.

Using Pull-Down Menus

PROJECT
New
Load
Save
Save As
Defaults
Print
Present . . .
Runtime
Video Setup
About
Quit

EDIT
Collect
Copy
Info
Preview
Telescope
Search

TOOLS
Object Editor
Videodisc
Database

Pull-down menus are lists of choices that drop down from the title bar of the AmigaVision screen as in other Amiga programs. To reveal them, press and hold the right mouse button. This causes three menu titles to appear at the top of the screen. To open one of the menus, continue to hold the right mouse button and move the pointer to the menu title. To make a choice from the menus, move the mouse pointer (keeping the button down) to the desired choice. When the option you want is highlighted, release the mouse button.

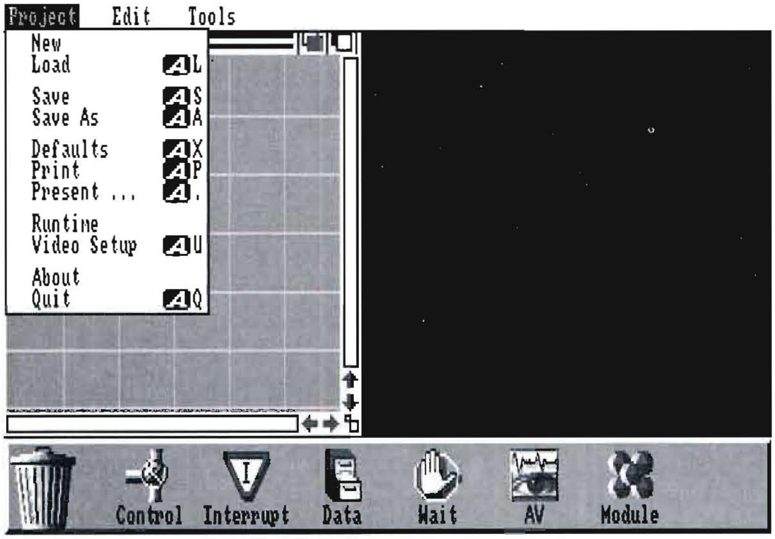
Some menu selections have *keyboard shortcuts* printed next to them. For example, you may bypass using the mouse to access the Collect option by holding the right Amiga key down while pressing the "O" key.

If a pull-down menu option is not currently applicable, it is ghosted (looks fuzzy), and cannot be highlighted.

There are three pull-down menus in AmigaVision:

- **Project Menu.** Used to configure, create, save, and present your AmigaVision projects.
- **Edit Menu.** Used to modify the current project.
- **Tools Menu**. Opens the Object Editor, the Videodisc Controller, and the Database.

The Project Menu



Each selection in the **Project Menu** is summarized below.

New

The New option has two submenu items: Flow and Content. When you choose one of these submenu items, AmigaVision opens a window which you can use for editing or creating an application. If there is an open window on the screen, the new window will open on top of it. To move a window, click on the Drag Bar gadget and drag the window to the new location. The Flow and Content windows are described later in this chapter.

Load

The Load option opens an application you created and saved earlier.

After you choose the option, the File requester is displayed. You specify the volume, drawer, and name of the file to open.

Save

The Save option saves the changes using the current filename of the application you are editing.

If you have already named the file, AmigaVision stores the application under the filename you previously specified. If you are editing a new application and it is not yet named, AmigaVision will prompt you for a name. After you save the application, it remains on the screen until you close the window.

If the file is an application flow, then the extension '.AVf' is added to the filename when saved. For content files the extension '.AVc' is added.

The Save option will also create a backup of your application if the BACK option is set. See Chapter 1 for information about setting AmigaVision options. The backup file has the same name but with a '.bak' extension. The backup will be updated each time you save more changes to the original application.

Save As

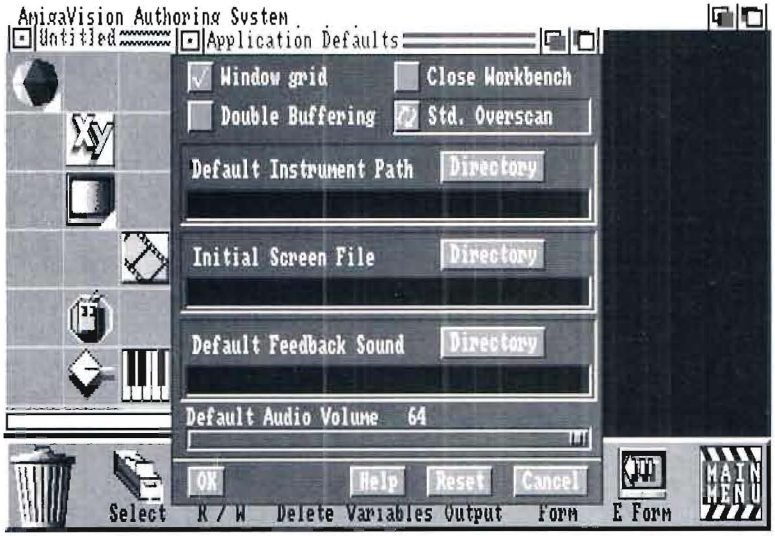
The Save As option saves a new application or saves the changes you have made to the current application under a new filename.

After you choose the option, the File requester is displayed. You specify the volume, drawer, and name of the new file.

Defaults

The **Defaults** option specifies for each application a set of default parameters that will be saved with the application. If you are editing different applications in different edit windows, you can specify separate defaults for each of the windows.

Click on the Edit Window first and then select **Defaults** from the pull-down menu. You will be presented the **Defaults** requester.



AmigaVision Authoring System

Application Defaults

☑ Window grid
□ Close Workbench
□ Double Buffering
☑ Std. Overscan

Default Instrument Path

Initial Screen File

Default Feedback Sound

Default Audio Volume 64

If you want a grid displayed on top of your edit window, you can turn on the *Window Grid* gadget. This grid can be helpful in giving visual alignments when placing icons in the outline.

Turn on the *Double Buffering* gadget ON if you want to remove the flickering associated with cookie-cut brush objects.

You can also choose between "Standard Overscan" (Std. Overscan) or Maximum Overscan (Max. Overscan). Overscan is used to eliminate "dead space" around the borders of a picture.

Set the *Close Workbench* gadget to the ON state if you want to save memory by closing down the Workbench screen during a presentation. The Workbench screen will be closed only if no other application is using the screen. For example, if you have a CLI window open, AmigaVision will not be able to close the Workbench.

There are four fields for specifying defaults:

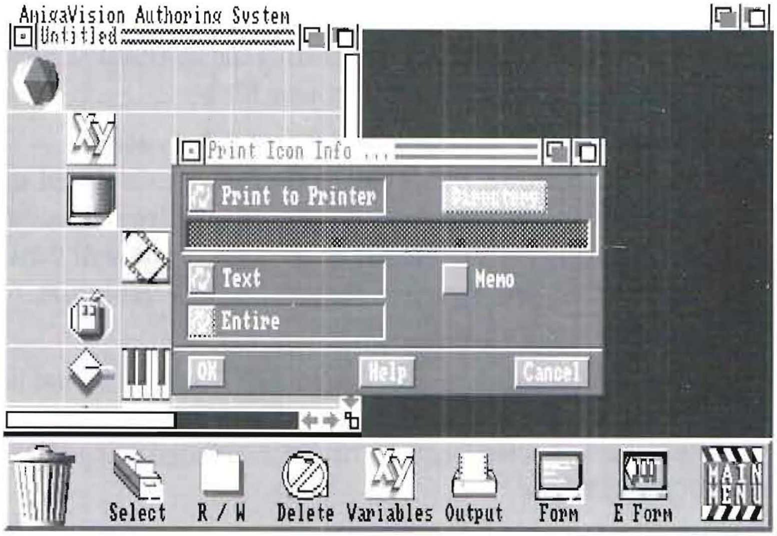
1. **Default Instrument path.** The Default Instrument Path specifies the path name for locating the instrument files used in playing back SMUS music files.
2. **Initial Screen file**. The Initial Screen File is used to select a picture file that will be the first one displayed at the start of a presentation. It is a good practice to specify an introductory picture file or the application will start off with a blank screen in the default high-resolution mode with the four Workbench colors.
3. **Default Feedback sound**. The default Feedback Sound is the name of the digitized sound file that you want to playback as a feedback for hit boxes created using the Object Editor.

4. **Default Audio volume.** The default Audio Volume is an index in the range of 0 to 64 for specifying the default volume of all the sound files: digitized audio, music, and speech. In the icons that control these sounds, you have the option of overriding the default setting. This option cannot be used for controlling the audio from the two channels of videodisc players, which do not have any volume control through software. The volume control feature helps you select a volume for sounds from the computer that is consistent in loudness with the audio from the videodisc player.

Print

The **Print** option prints out the information in the outline of your application in the currently selected edit window.

When you select this option, the *Print* requester is presented.



Click on the *multi-state* gadget to select the output destination: **Printer** or **File**. If you choose **File**, type the name of the file in the filename field or click on the *Directory* gadget to open the File requester. Here you may choose a filename or create a new one for the output. If you choose the Printer option, make sure that a printer is connected to the Amiga computer, it is turned on, contains paper for printing, and is on-line.

If you choose the *Printer* option use the multi-state gadget to specify whether you want Text or icon Image printed. If you select *Text*, the name field will be printed. If you select *Image* the icon images will be printed. In both cases the printing will be in the same flow sequence as seen in the Flow Editor. If you choose the *File* option you can print to either a file or a printer. You cannot print images to a file.

You may print either a selected icon or the entire outline. To print one icon, select it before choosing the *Print* menu option. If there are no selected icons, AmigaVision prints the entire outline.

The names of the icons will be printed. If you wish to print the memo fields as well, set the *Memo* gadget to ON state by clicking on it.

When you click on OK, AmigaVision starts printing.

If you want to abort printing and exit the Print requester, click on Cancel.

Present . . .

The **Present** option runs the application.

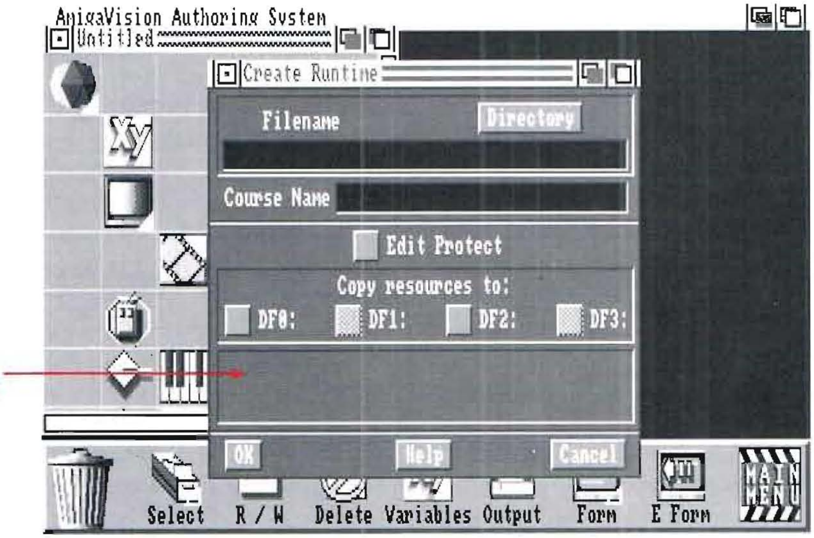
First select the window containing the application you wish to run by clicking in the appropriate flow window. Then choose **Present** from the menu. AmigaVision then clears the edit screen and begins the presentation. To abort the presentation, click on the right mouse button. If you abort a presentation, you will be returned to the edit screen at the point where you aborted and the icon where you aborted will be displayed as selected.

Runtime

The **Runtime** option has two submenu items. **Create** and **Install**. The **Create** option moves a finished application to disks for distribution and use. The **Install** option installs an application from the source to a destination file.

An AmigaVision application consists of the flow file and any associated files necessary for it to run the presentation, such as the audiovisual files, the database files, fonts and musical instruments. The **Runtime** options help install these files.

When you select the **Create** option, the *Runtime* requester is presented.



status window

1. ***Click on the Directory gadget to open the File requester. Then, specify the name of the application from which you wish to create a runtime version. The name will appear in the Filename field.***
2. ***Next, click on the Course Name field. Type in the name you wish to give for the new runtime version.*** This does not need to be the same as the Filename. The Course Name is used to label the floppy disks and the flow file.
3. ***Select Edit Protect if you want to protect the application from editing.*** The user will be able to run the presentation but will not be able to view the icon structure in a flow edit window.
4. ***Specify the drives for copying the application to the disk (DF0-DF3).***

The Status Window tells you what files are being copied and to what disk they are being copied. It also informs you of any problems encountered while copying the files. Your disks must already be formatted beforehand.

If one disk becomes full during the Runtime module creation, AmigaVision will prompt you for the next disk. The disks will also be labeled in the order in which they are copied.

You cannot install more than one application on the same disk because the disk labels are marked by the name of the application.

When the **Runtime** option creates runtime disks, it copies all audiovisual and database files referenced in your application. It installs them in proper directories and modifies the application to reference the new pathnames and directories.

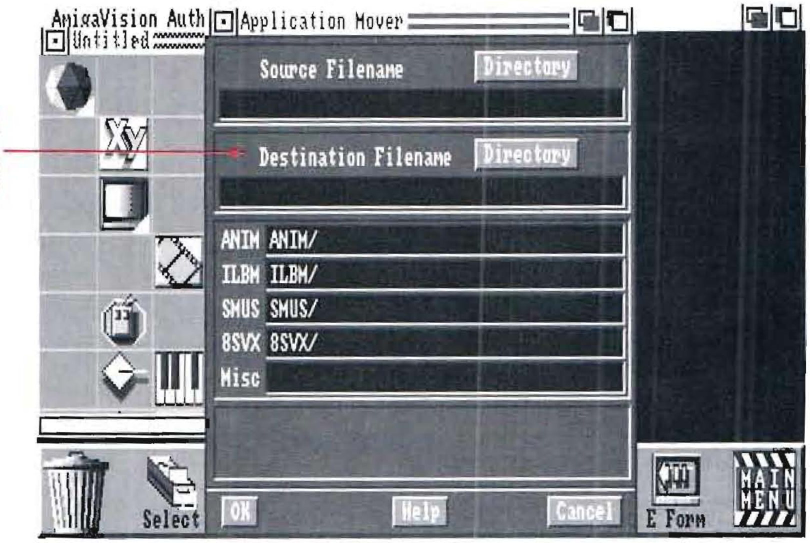
If a filename is specified as a variable, then the Runtime option does not copy a file. It simply creates a Log file on the runtime disks. The Log file also consists of a list of instruments for musical playback and any non-standard fonts referenced in your application.

Since some instruments and fonts are proprietary, you need to be careful copying these into an application intended for distribution. You are responsible for distributing any copyrighted material.

Please read the Log file using any word processor editor, or by the TYPE command of the Shell for a list of all the non-installed resources, such as picture files, instruments and fonts. The Log file is in ASCII format.

In the case of instruments, you can install them on the runtime diskettes by loading them in your application using a **Resources** icon (refer to Chapter 6). The **Create** option will then install these instruments on the Runtime disks rather than listing them in the Log file. For a user to be able to double-click on an icon to execute, the system needs to know where AmigaVision is located. This can be accomplished by either specifying it in the tool type (info from the Workbench menu) or by assigning AmigaVision: Work: AmigaVision as mentioned in Chapter 1.

The **Install** option installs an application either from a floppy disk to your hard disk or from one drawer in the hard disk into another drawer. When you choose this option the *Application Mover* requester is presented. You must specify the application source filename, the destination filename, and the specific paths for animation, ILBM (picture), SMUS (music), 8SVX (digitized sound), and other miscellaneous files. Your application will be installed in the destination directory. Your flow file will be automatically modified to reflect the changes in file directories. Using this you can move applications between disk drives and/or between directories.



choosing "install" opens this requester

Application Mover

Source Filename

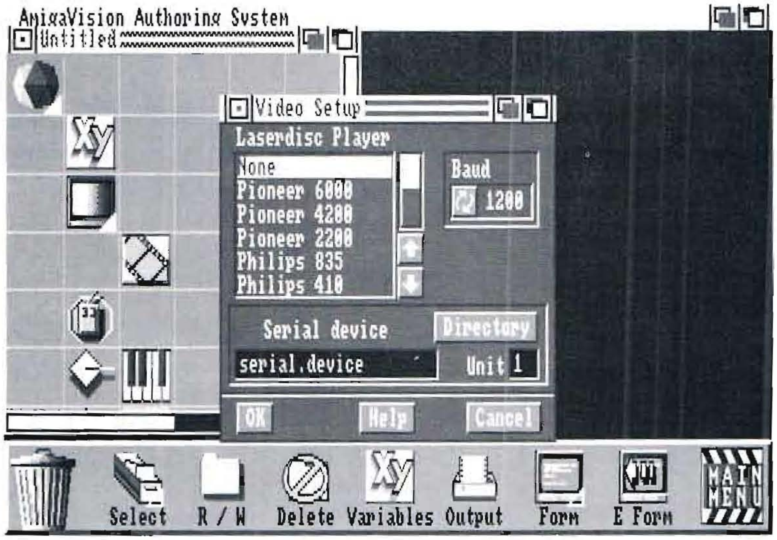
Destination Filename

ANIM ANIM/
ILBM ILBM/
SMUS SMUS/
8SVX 8SVX/
Misc

Video Setup

The **Video Setup** option allows you to specify the hardware you will be using.

When you select this option, the Video Setup requester is presented.



Video Setup

Laserdisc Player
None
Pioneer 6000
Pioneer 4200
Pioneer 2200
Philips 835
Philips 410

Baud 1200

Serial device serial.device

Unit 1

1. ***Click on the video device from the list of devices displayed.*** It will be highlighted after selection.
2. ***Click on the Multistate gadget and select the Baud Rate from 1200 to 9600.***

   Some videodisc players, such as the Pioneer 4200, have baud rates limited to a maximum of 4800. Please make sure the baud rate on your player (typically set using dip switches in the back of the device) is consistent with your selection in this requester; otherwise the videodisc player will not operate correctly.
3. ***Click on the Directory gadget to specify the serial device driver name.***

   If you use the standard driver, serial.device, supplied with your Amiga, it will be in the DEVS: drawer, and you won't need to adjust it. If you are using a multi-port serial card, you must specify the name of the device driver file in the device field.
4. ***If you are using a multi-port serial card, there is a field provided for Unit where you can type in the number of the logical device unit number of the serial port. If you are not using a serial card, the Unit number is 1.***
5. ***Click on OK.***

   Video Setup copies the Player.device, creates a Players subdirectory, copies the driver for your particular player, and puts them all in your DEVS: directory.

About

If you choose the **About** item from the menu, a window displays the version number and copyright dates of the AmigaVision program you are using.

Quit

The **Quit** option ends your editing session and exits AmigaVision.

After you choose the **Quit** option, a requester prompts you to save any changes you have made. If you have been editing in multiple windows, then for each window which was modified, you may choose:

Save — save changes to the application under its current filename and then exits AmigaVision to Workbench.

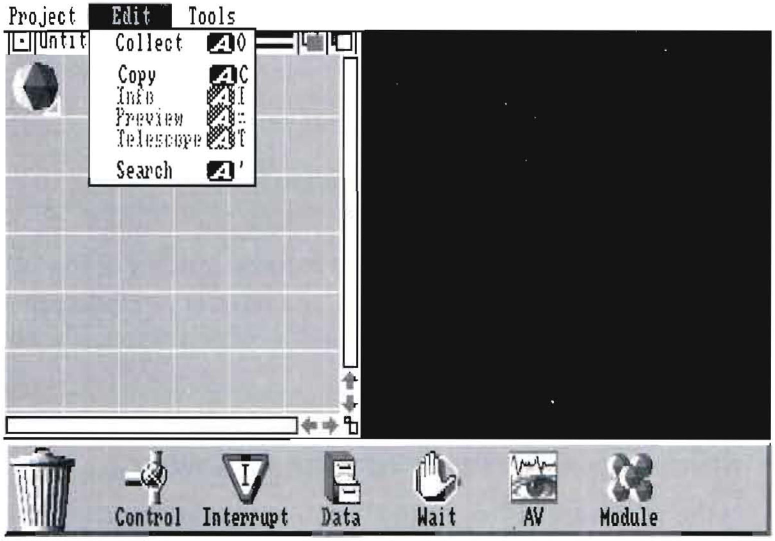
If the work is untitled, AmigaVision opens the File requester where you will specify the volume, drawer and name of the work to be saved.

When there are no remaining windows that have been modified, AmigaVision exits.

Close — exit AmigaVision to Workbench and do not save changes.

Cancel — do not exit but return to the editor.

The Edit Menu



Collect

The **Collect** option allows you to gather a group of icons into a **Module** icon or **File Folder** icon. The collection process is used to create a parent module for a group of icons. When using the Collect function only **Module** icons in the Flow Window and the **Folder** icons in the Content Window will be created as parents for a group of icons.

To Collect a group of icons:

1. ***Select the Collect option from the pull-down menu.***

   A pointer with a small rectangle attached to its end is displayed.

2. ***Position the pointer on one of the corners of the group of icons to be collected. Press and hold the left mouse button.***

   A small rectangular "rubber band" appears in the window. This rubber band is sized by moving the mouse.

3. ***Drag the mouse to expand the rectangle, including all the icons that you want to collect.***

   The window will scroll up or down to let you to collect as many as you wish.

4. ***Release the button.***

   A requester will ask if you wish to collect the icons in the rectangle.

5. ***Click on the OK gadget.***

   A Module icon will appear as a parent to the collected icons.

The Collect option stays in the collect mode until you select the **Collect** option again from the pull-down menu to turn it off.

Copy

The **Copy** option allows you to copy a selected icon from one place to another within the same window.

1. ***Select the Copy option from the menu.***

   A pointer with a small rectangle attached to its end is displayed.

2. ***Drag the icon you want to copy to the new location, and release the button.***

3. ***An identical copy of the icon and its contents are made at the new location. The Copy option, once selected, stays in the copy mode until turned off by a second selection of the same.***

No menu option is needed when copying an icon from one window to another. Simply drag the icon to the new window and release the button.

When the **Copy** option is disabled, you can drag icons from one location to another within a window.

When you collect, copy or move an icon to a location that is not currently visible through the Edit Window, you can hold the left mouse button and move it either to the top or the bottom of the window and then press the right mouse button to scroll the window. You may also use the directional arrows to scroll the window.

Info

The **Info** option allows you to define an icon you have selected.

1. ***First select the icon you wish to define. (Click once on the icon with the left mouse button.)***
2. ***Then, choose the Info option from the Edit pull-down menu.***

   A requester prompts you for a name, memo (i.e., remarks), and other relevant information which will be associated with the icon.

3. ***When finished, choose the OK gadget to save the specifications and exit the requester or choose the Cancel gadget to close the requester and exit the requester without saving the changes.***

Preview

The **Preview** option executes an icon. If the icon is a parent, its children will be previewed as well.

First select the icon to be viewed. Then, choose the **Preview** option from the **Edit** pull-down menu. AmigaVision clears the screen and presents the icon as it will appear in the presentation.

To exit Preview mode, click the right mouse button.

Telescope

The **Telescope** option collapses or expands the selected children or parent icons. This option only has an effect on icons containing children.

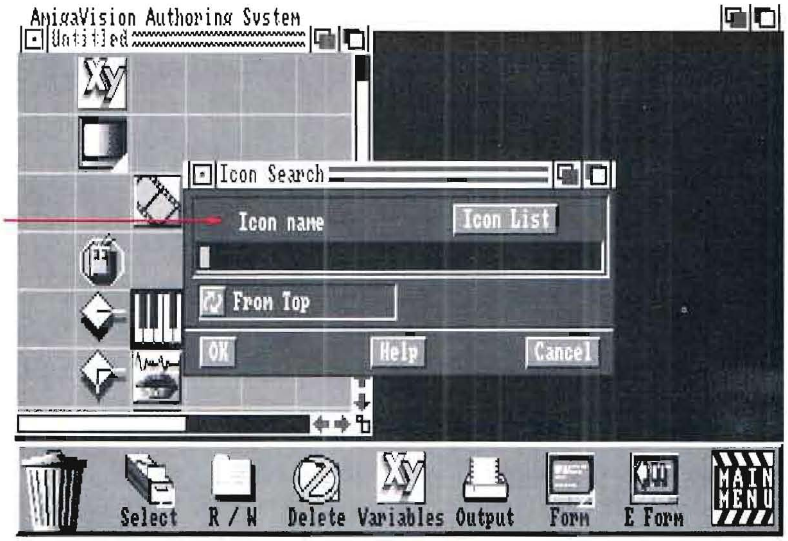
1. ***Select the parent icon to be collapsed or expanded.***
2. ***Choose the Telescope option from the Edit pull-down menu.***

   If the selected icon is a parent with children attached, the children icons will be collapsed so that only the parent icon is visible in the outline.

If the selected icon is a parent icon with collapsed children, the **Telescope** option will expand it to show all of its children.

Search

The **Search** option helps you locate an icon by name. When you choose this option, a requester is presented. You can click on the Icon List gadget to get a list of all the icon names in that window. You can select a name and the Flow Window will display icons starting from that icon. Also, that icon will be selected.



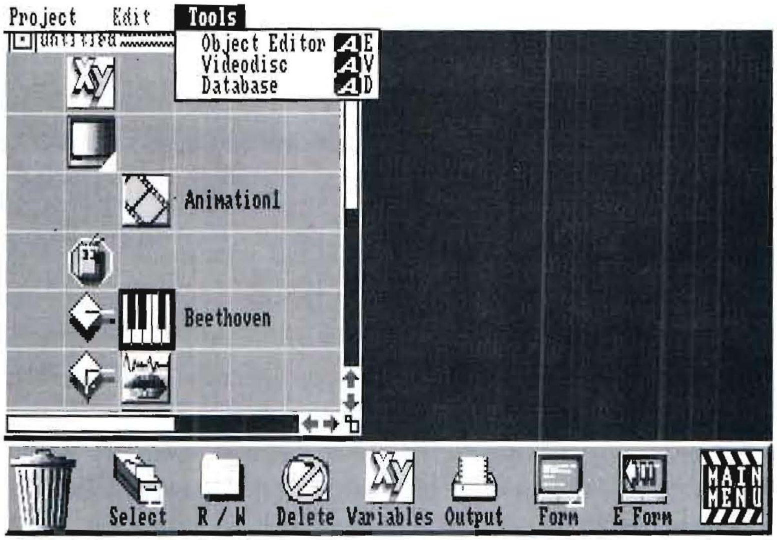
choosing "search" opens this requester

Typing the name of the icon in the field bypasses the list. In this field you can use the wildcard characters * and ?. If you do not remember the exact name of the icon, use ? to represent a single character and the * for one or more characters. For instance, if you vaguely recall the icon name as starting with "St" and ending with the last letter "r", then you can type in the field a name such as "St*r". This will locate an icon if the name matches even with the unknown characters.

You can use the *Multi-state* gadget to specify the location in your flow from which to start the search. You can choose *From Selected* or *From Top*. The former one will search from the currently selected icon, and the latter will search from the starting icon in the application.

Tip: It is a good practice to name the icons as soon as you use them. This way you can locate various icons in a large presentation very easily. The **Search** option will not work without icon names.

The Tools Menu



Object Editor

The **Object Editor** option opens AmigaVision's interactive screen editor. This editor is used to place display objects and text onto the screen.

After you choose the option, AmigaVision clears the screen and enters the editor. The **Object Editor** is a separate utility and has a new set of pull-down menus. To see the editor menu choices, press and hold the right mouse button near the top left corner of the screen. The **Object Editor** is also entered through the requesters of some of the icons. For more information on the use of the **Object Editor**, see Chapter Four.

Videodisc

The **Videodisc** option opens AmigaVision's interactive videodisc controller. You can use it for browsing through video frames.

After you choose the option, the **Videodisc Controller** will open on the screen. The **Videodisc Controller** is also entered through the requester for the Video icon. For more information on the use of the **Videodisc Controller**, see Chapter Four.

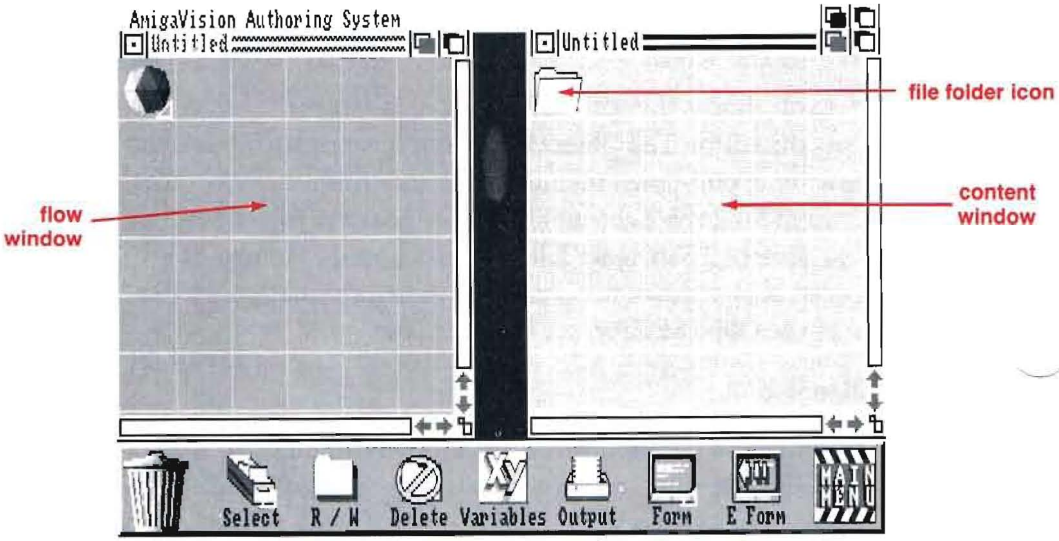
Database

The **Database** option opens AmigaVision's database module.

After you choose the option, the database window will open on the screen. For more information on its use, see Chapter Four.

Using Flow and Content Windows

AmigaVision has two types of editing windows: **Flow Windows** and **Content Windows**. The window displayed when you start AmigaVision is a Flow Window. This window is where you will create and save projects for later presentation; it is identified by a **Module** icon.



Content Windows are used for creating and saving libraries of multimedia elements. These are the audio-visual elements such as graphics, sound, music, video segments and text. You can think of the Content Window as a place to store audio-visual "parts" for use in the programs you create in the **Flow** windows. The **Content** windows are identified by a **File Folder** icon.

Although the audio-visual icons in a **Content Window** cannot be run as a program, you may open the requesters of the icons in the normal way and also use the *Preview* function, which performs the action of that icon so you can see how it looks before exiting the requester.

From the Project pull-down menu, select *New* and either *Flow* or *Content* to open the appropriate window type. Icon placement and editing are identical in both windows; however the following rules apply:

- You may place only icons from the Audio Visual submenu in the **Content Window**.
- You may create a new File Folder by "collecting" a group of icons in the Content Window. To do this you choose Collect from the **Edit** pull-down menu and draw a rectangle around the icons you wish to group.
- You may also copy icons from one window to another.
- You may have multiple **Flow** and **Content Windows** open simultaneously.

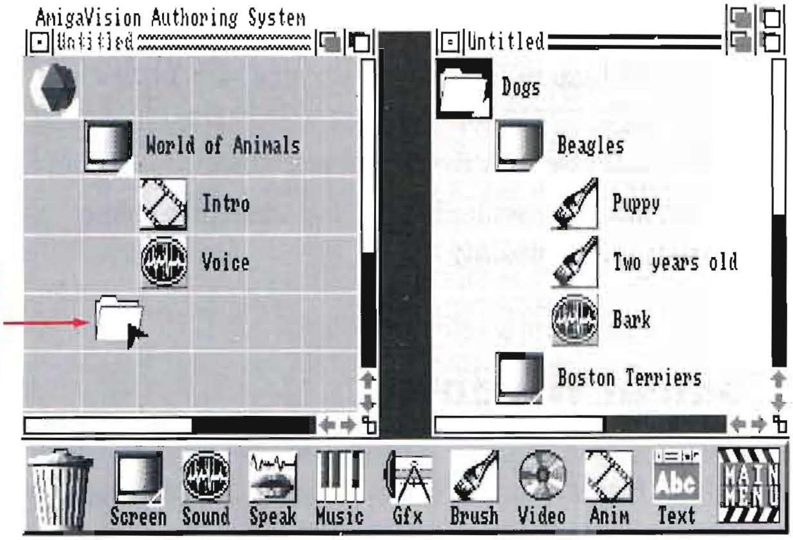
Content Windows

Content Windows appear in a lighter shade of gray and begin with a **File Folder** icon. File Folder icons make organizing all of your audio-visual elements easier for multiple projects.

As an example of organizing audio-visual elements, we'll use a collection of related graphics, animations and sound. The File Folder serves as a way to define where all of these elements are stored so that when you design your projects you don't have to worry about anything other than the name of the element you want.

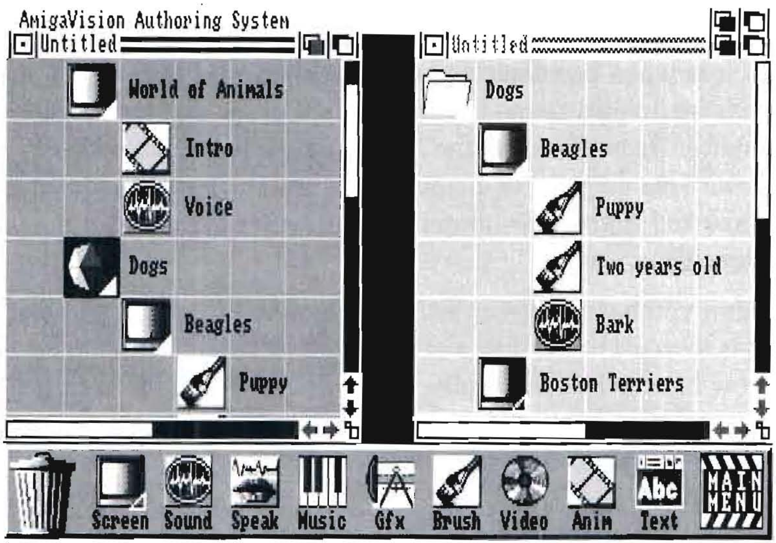
In our example, all of the icons are stored in one file folder with a separate screen for each dog. By identifying each picture and sound file with a recognizable name, you can then simply drag icons from the Content Window to the Flow Window and have all the information already set with regard to location of the picture file.

You may drag individual elements out of a Content Window, as well as the entire file folder. This way the same information can be used in multiple presentations without redefining all the elements. To include all the elements of the *Dogs* folder, you would drag the folder into the flow at the point you wanted to insert it.



the dogs folder is dragged into the flow window

After you drop the folder icon in the Flow Window it turns into a Module icon with all its subsequent child icons and is inserted into the flow. After dropping the folder your flow would look like this:



All the elements from the Content Window are now part of your flow and can be further modified without affecting the versions stored in the Content Window. Once such sections are organized, you can easily interchange them in multiple flow windows. A perfect example is a videodisc where certain sections are always in one place. Instead of remembering that the section on, say "Training Your Dog," is on frames 1300 through 1456 and "Washing Your Dog" is on frames 1789 through 2389, you could create a File Folder with nothing but the different sections of a videodisc in it, allowing you to drag the "Training Your Dog" icon into your flow instead of re-keying the frames every time you wish to use them in different presentations.

Later you may wish to only use the "Beagle" section of the File Folder in another presentation. Instead of redefining every graphic and sound file, you would open your "Dogs" Content Window and drag the "Beagle" screen icon into your new flow. Instead of grabbing the File Folder, you would pick up the Screen icon.

Just as when you dragged the File Folder icon, the Screen icon you are dragging would be highlighted in the Content Window and would be placed in the Flow Window just as any other icon. This dragging of elements can be done between multiple Flow and Content Windows that are open on the screen at the same time.

After you have dragged the Screen icon to your Flow Window, it is inserted in the flow along with all its child icons. You have now reused the same audio-visual elements in more than one presentation but only had to define filenames, transition types and other parameters once. This is the power of the Content Window.

Chapter 4

Editors and Tools

Chapter 4: Editors and Tools

AmigaVision has several editors and tools you can use to build your applications, including:

- **Expression Editor** — for creating and altering variables and expressions
- **Object Editor** — for creating and modifying display objects
- **Videodisc Controller** — for browsing and selecting from a videodisc player
- **Database Editor** — for creating and editing database files

These topics are discussed in this chapter.

Expression Editor

The Expression Editor is used to define *variables* and *expressions*. Variables are useful for storing values in either numerical or in alphabetical (string) form. Variables can then be used in expressions.

Variables

Any alphanumeric string of characters can be used to name a variable, but the name must begin with a letter.

There are four types of variables available in AmigaVision. These types are similar to the variable types in most computer programming languages:

1. ***String.*** To store alphanumeric character strings.
2. ***Numeric Integer.*** To store integers.
3. ***Numeric Floating Point.*** To store numbers with decimal points (floating point).

   Exponent notation (1E6, etc.) is not supported for numeric variables.
4. ***Boolean.*** To store either TRUE or FALSE states.

Expressions

Expressions specify the value of a variable, the relationship between variables, or conditions. There are two general types of expressions:

1. ***Assignment expression:*** Score = 100 is an example of an assignment which is interpreted by AmigaVision as assigning a value of 100 to the variable called Score. Assignments can also be made so that one variable is defined in terms of others. For example:

   Score = X * 100.

   String assignments are made as: Name = "John"

   A Boolean assignment would be: Status = TRUE

2. ***Conditional expression:*** Score = = 100 or Score >= 100 are conditional expressions. These are interpreted by AmigaVision as "if Score is equal to 100" or "if Score is greater than or equal to 100." String comparisons can be stated as: Name = = "John" which translates to "if the string variable Name has a current value equal to John." Do not use "if" when you state conditional expressions as it is implied already by the use of conditional expressions.

Creating Variables in AmigaVision

You can create new variables in AmigaVision simply by defining them. You define variables in the Expression Editor by using the new variable name on the left side of any expression followed by the operator = which in turn, is followed by either an expression or a constant. For instance, the following statements are legitimate definitions:

name = "JOHN"	(a string variable)
G = 6.6	(a floating point numeric variable)
x = y * 10	(a numeric integer variable)
switch = TRUE	(a Boolean variable)

Note that the = operator in this context works as an assignment operator, not a numeric operator.

You can even define variables implicitly inside of a conditional expression. For example, the following definition is allowed:

(x = y * z) >= 10 (a conditional expression)

You can define variables using three icons in AmigaVision:

1. ***Module icons***
2. ***Subroutine icons***
3. ***Variables icons***

Module and Subroutine icons are only used to create local variables. Local variables are temporary variables that retain their definitions until the actions of all the children of the Module or the Subroutine icons are performed. You can reuse local variable names in other areas of the flow diagram by using new definitions after the completion of the Module or Subroutine icons where they were defined earlier. You cannot use the **Module** or **Subroutine** icons to modify variables, only to create them.

In contrast, the Variables icon can be used to create only *global* variables. Global variables are permanent variables that retain their definitions from their point of definition through the rest of the application. Unlike the local variable names, the global variable names cannot be reused, as new definitions of the variables are simply assumed to be modifications of the original definitions.

Variables can also be modified at any point in an application. Whether a variable is local or global, you can modify its value only through the **Variables** icon. You cannot use the **Module** or **Subroutine** icons to modify variables, only to create them. If you try to modify a variable value in a **Module** or a **Subroutine** icon, it is assumed that you are creating a new local variable.

If you want global variables for use in an entire application, then you must define them using the Variables icon as the very first icon in your application flow. When you find a need for adding new global variables, you can simply add to the list of the variables in the very first icon. This keeps all your global variable definitions organized in one place.

It is a good practice to adopt variable naming conventions that set apart global from local variables. For instance you can name all global variables with a prefix of "G" and all local variables with a prefix of "L."

When defining a new variable or an expression, you can use all the currently available variables. For instance the expression:

```
E = A * B * C + D
```

assumes that the variables A, B, C and D are already defined. They can be either local or global variables.

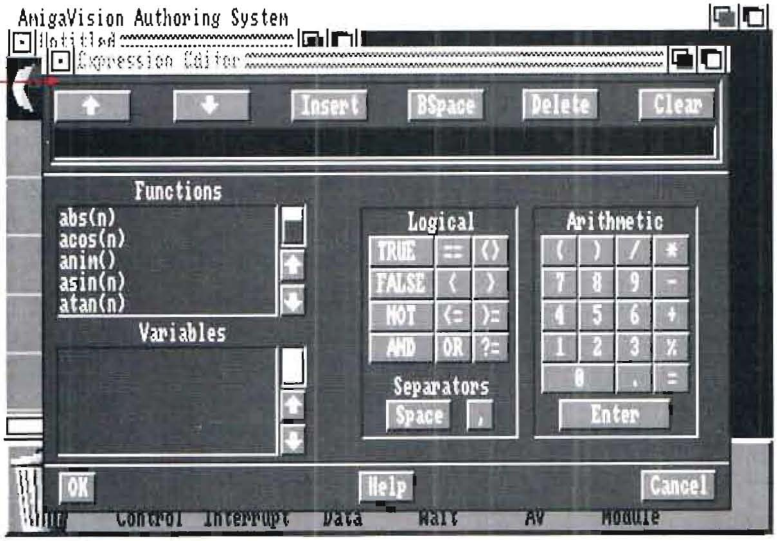
When you want to define a variable in terms of other variables, the other available variables will be displayed at the lower left list in the Expression Editor. This list includes all global variables defined by the **Variables** icon and all valid local variables defined by a **Module** or a **Subroutine** icon prior to that point in the flow. Variables created inside of a conditional expression will not be displayed in the list.

The variable list in the Expression Editor does not display any local variables that are undefined prior to that point in the flow. Thus, the list is a useful indicator of the variables that you can use in defining other variables or expressions.

Using the Expression Editor

You enter the Expression Editor in one of four ways:

1. *From the Module, Subroutine and Variables icon requesters, you click on the Insert gadget or double-click on an expression shown in the expression list in the requester.*
2. *From the If-Then, If-Then-Else, and Conditional Goto icons, you double-click on the icon in the flow.*
3. *From the Loop icon, if you choose either the Conditional (Test at Start) or the Conditional (Test at End) options, then click on the Expression Editor gadget in the requester. It works in a similar way to the Wait for Condition icon.*
4. *From the* Input Field Definition *requester in the Object Editor.*



the expression editor

You are allowed to create multiple expressions for **Module, Subroutine** and **Variable** icons by using the up and down arrows which appear in the Expression Editor when you have entered it through one of these icons. For example, if you want to initialize the values of five different variables, such as Name, Age, Height, Weight and Telephone, you do not have to use five separate Variables icons. You can specify: Name = "John", Age = 30, Height = 60, Weight = 150, and Telephone = 5551212, by placing one **Variables** icon in your Flow Window and defining the variables through the Expression Editor with the up/down arrows.

The Expression Editor is divided into five separate regions:

1. ***Functions.*** Lists standard arithmetic and string functions supported in AmigaVision.
2. ***Variables.*** Lists all the local and global variables that are available for use.
3. ***Logical Operators.*** Used exclusively in conditional expressions.
4. ***Arithmetic Operators.*** Used in assignment expressions and as a general-purpose numerical pad.
5. ***Expression Field.*** Used for entering or editing an expression.

You define variables using expressions in the expression field near the top of the Expression Editor.

You may either type in an expression using the keyboard or build it with the mouse by selecting the desired items on screen from the gadgets, variables and functions in the Expression Editor. For example, you might create a variable called SCORE by typing SCORE = 100 in the expression field or by typing SCORE and clicking on the = gadget and then on the gadgets representing the numbers 1, 0, and 0. Characters and expressions are entered into the expression field at the location of the cursor.

When you have completed assigning a value to a variable, click on the Enter gadget or press the Enter key on the keyboard. In the "SCORE" example above, since a variable by the name SCORE does not yet exist, a new variable is created and displayed in the Variables Window.

A single-character code **I**, which stands for integer variable, is also displayed with the new variable SCORE. The other codes are **B** for Boolean, **S** for string, and **F** for floating point.

Arithmetic Operators

The Arithmetic operators in the Editor follow the standard symbols in most programming languages.

+	for addition
–	for subtraction
*	for multiplication
/	for division
()	to control the order of operations.
%	stands for modulus. For example, the expression
	a = 16 % 3
	will assign a value of 1 for a, which is the remainder when you divide 16 by 3.
=	only used for assignment of values or expressions.
	The *Enter* gadget simulates the keyboard Enter key.

Logical Operators

The Logical operators are used for specifying conditional expressions. Use AND for conjunctions, OR for disjunction, NOT for negation, < for less than, > for greater than, <= for less than or equal to, >= for greater than or equal to, <> for not equal to, and = = for testing the equivalence of two values.

String Operators

In AmigaVision all string comparisons using the = = symbol are insensitive to the case of the characters in the string. In other words, all strings can contain mixed upper or lower cases. If you need to convert all the characters in a string to upper or lower cases, there are special string functions available for this purpose.

The **?=** is for matching patterns in a string. For example, you have a string variable called ANSWER which contains the string that was entered by a user in response to the question: "Who wrote *The Sound and The Fury*?" If you want to see if the word (substring) "Faulkner" appears in the ANSWER, then you will use the conditional test expression:

ANSWER ?= "*Faulkner*"

This test will return a TRUE value if ANSWER contained the word "Faulkner" at any location, even in a long sentence.

The pattern string you specify can contain two wildcard characters: the * character to indicate any phrase (including spaces) and the character ? for any single character. These wildcard characters are very useful in specifying spelling and grammar tolerance levels. In the above example, for instance, if you also want to forgive poor spelling of the word "Faulkner" then your pattern matching expression will be something like: ANSWER ?= "*Fau?ner*" Thus even an answer such as

"Faukner" will be considered as a good answer for the question "Who Wrote *The Sound and The Fury*?"

Use the *Space* gadget to insert a space for enhancing the readability of an expression; use the , (comma) gadget for separating the parameters in a function. The available functions are described later in this section.

Expression Field Gadgets

The Expression Field has six gadgets associated with it.

- **Up arrow**
- **Down arrow**
- **Backspace**
- **Delete**
- **Clear**
- **Insert**

The up and down arrows are for access to the list of expressions. These two arrows are disabled if you access the Expression Editor through a condition such as the **If-Then** icon. For every conditional expression only one entry is permitted. The *Backspace* gadget is useful for deleting the character to the left of the cursor inside the Expression Field. The *Delete* gadget is for deleting the character right under the cursor and the *Clear* gadget is for clearing the entire field. The *Insert* gadget inserts the expression into the scroll list.

When you complete the definitions of variables or expressions, click on the *OK* gadget at the bottom left corner of the Expression Editor. If the expression you entered has invalid syntax, an error message is shown in a message window. If the expression is valid, the Expression Editor saves your expressions. From that point on, any time you enter the Expression Editor through that same icon, the Expression Editor will display the previously defined and currently valid variables for editing.

To exit the Expression Editor without saving, click on the *Cancel* gadget. This exits the Expression Editor without saving your current definitions.

Functions

You may also use AmigaVision's library of functions to create expressions. An example of an expression using a function is

response( )= = "box 1"

where "box 1" is a response string assigned to a hit box created in the Object Editor. The response function returns the current value of the response string from a hit box or the keyboard. Functions can appear anywhere in an expression. The following is an example of a valid expression using functions:

NewValue = max(a, min(b, c))

Here the minimum of *b* and *c* is found first; the maximum of that value and *a* is found next, and then the result is assigned to the variable NewValue.

AmigaVision functions are shown in the Functions Window of the Expression Editor. To view the complete list, click on the up or down arrows at the right side of the window. In each function, string variables are represented by "s," "s1" and "s2"; literal numeric variables are represented by "n," "n1," and "n2," and variables or numbers by "v."

Mathematical Functions

abs(n)	absolute value of n
acos(n)	arccosine of angle of n degrees
asin(n)	arcsine of angle of n degrees
atan(n)	arctangent of angle of n degrees

cos(n)	cosine of angle of n degrees
exp(n)	the exponent of n
inc(v)	increments variable v by 1
integer(n)	integer value of numeric n or string "n"
float(n)	floating point value of numeric n or string "n"
log(n)	natural logarithm of the number n
log10(n)	logarithm to the base 10 of the number n
max(n1,n2)	returns the larger of two numeric values n1 and n2
min(n1,n2)	returns the smaller of two numeric values n1 and n2
pow(n1,n2)	finds the n2th power of n1
sin(n)	sine of angle of n degrees
sqrt(n)	finds the square root of the number n
tan(n)	tangent of angle of n degrees

String Functions

findstr(s1,s2,n)	find s1 in s2 starting at character n
strcat(s1,s2)	concatenates s2 to the end of s1
string(n)	convert to string the numeric n
strlen(s)	get the length of string s
substr(s,n1,n2)	the substring of string s beginning in position n1 of length n2
tolower(s)	converts all the characters of the string s into lower case letters
toupper(s)	converts all the characters of the string s into upper case letters

Special Functions

anim( )	returns current frame number of the animation
boolean(v)	returns TRUE if variable v is non-zero, "T", "TRUE", or "YES."
clock( )	the current time returned as a string (e.g. 23:03:15)
date( )	the current date returned as a string (e.g. 03/19/1990)
random(n1,n2)	returns integer in the range where n1 is lowest and n2 is highest
response( )	last response string entered from the keyboard or returned from a hit box
screenx( )	x coordinate of screen cursor location in current screen resolution
screeny( )	y coordinate of screen cursor location in current screen resolution
status( )	returns FALSE if a system request such as file access has failed
timer(n)	number (1 to 9) of timer, returns count in seconds to two decimal places
video( )	returns current frame number of the videodisc

You can click on a function and it will be inserted in the Expression Field at the location of the cursor. After insertion, if the function does not require any parameters, for example response(), the cursor will be placed to the right of the close-parenthesis. If it needs one parameter, then the cursor will be on top of the close-parenthesis, indicating that the function needs a parameter. If it needs two parameters, for example max(n1,n2), then there must be a comma between the two parentheses. Thus max(n1,n2) will appear in the Expression

Field as max() and you must type "10,7" to compare 10 and 7 Similarly, there must be two commas between the parentheses for three parameters and so on.

You can click on a variable in the variable list to insert it inside an expression in the Expression Field. Quite often you will use variables as parameters for the functions. A's mentioned earlier, you have to be careful not to use incompatible variable types. For instance, the function max(n1,n2) expects two numeric variables. Thus you cannot use string variables as parameters for max(n1,n2). The intricacies of using variables and expressions in the Expression Editor may take practice for new users to master.

Object Editor

You can use the Object Editor to create display objects for use in a course or presentation. Display objects are independent visual objects you can place on the screen as hit boxes, text, and graphic elements. The display objects are:

- **Rectangle**
- **Polygon**
- **Line**
- **Circle**
- **Ellipse**
- **Text**
- **Brushes**
- **Data Entry Fields**

Brushes are created using an external paint program such as DeluxePaint III. With the editor, you can create these shapes and turn them into user input areas that add interactivity to your applications. These input areas are called *hit boxes*. In AmigaVision, you can overlay these display objects and hit boxes on the screen, which may contain a background picture or video.

The following examples illustrate the various uses of the Object Editor:

1. ***Creating a menu of six choices from which the user may select one.*** To do this you might first create a picture in a paint package that details the visual appearance of the menu. Then you enter the Object Editor through a Wait icon (Wait Mouse, for example) and define six rectangular areas that define the regions on the menu for user choice. For each hit box you need to specify a unique string such as "A," "B," "C," "D," "E," and "F," respectively to represent the six choices. Then, you could use six If-Then-Else icons to check which response string was entered and specify the appropriate branching.
2. ***Creating interrupt options, such as Help.*** Suppose you want to provide context-sensitive help to the user at any point in an application. You decide to show a hit box called Help that users can click on to suspend the progress of the application and get help. To do this, you enter the Object Editor through an Interrupt icon (such as the Mouse Interrupt) and specify the appearance of the hit box. If the user clicks on this hit box, then you can set up useful help messages.
3. ***Creating toggle switches for the user to operate two-state switches.*** For instance, you might want the user to answer yes or no to a set of questions on the screen by clicking on hit boxes. You might further want the default answer to all the questions to be Yes and show the boxes in red color. If the user clicks on the boxes, you might want to turn the clicked boxes to blue indicating that the user has changed the state from yes to no. This gives useful visual feedback.

4. ***Creating digitized audio feedback to give a specific hint to the user.*** You might set up a hit box called Hint and expect the user to click on it when he or she needs a hint to answer a question. Then, when the user asks for a hint, you could provide it in a recorded voice message.

5. ***Creating a test for identifying parts of an object,*** such as a camera or an insect. This would be identical to a menu except that you may want to stay in the same picture until all parts are identified correctly by the user.

6. ***Creating textual feedback.*** You can use text strings as objects and specify feedback messages such as "Good" or "Try Again" to reinforce the users when they click on a hit box. Also, you can use the text objects to display the contents of specific variables that you have defined in your application. For example, you might want to have the name of the user displayed at the top right hand corner of the screen throughout the use of the application.

7. ***Creating a data entry form which allows a user to enter name, social security number and any other textual or numerical input.*** To do this, you enter the Object Editor through the Forms icon (a Database icon). You can specify input fields and also impose many restrictions on user input so that the form will take only acceptable inputs. You can link the input values to variables in your application.

As you can see, the potential uses of display objects are limited only by your imagination. By combining pictures, sound and display objects creatively, you can create a powerful and highly interactive applications.

Using the Object Editor

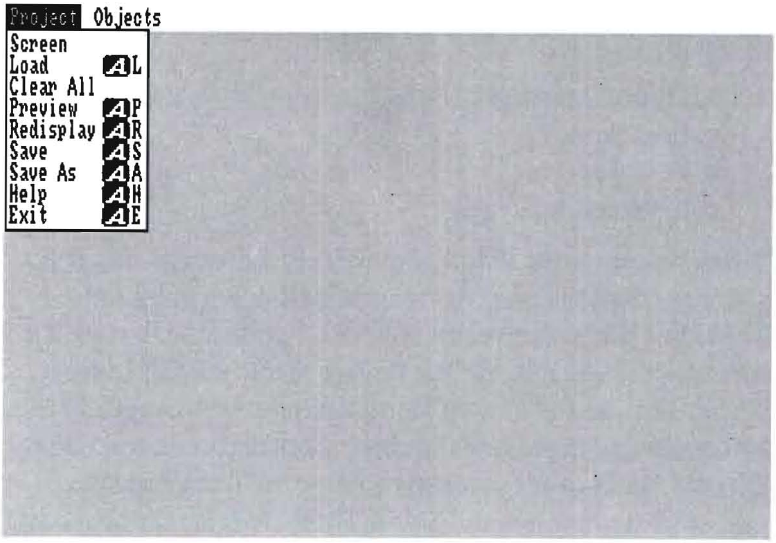
You enter the editor either by selecting Object Editor from the Tools pull-down menu or through one of the following icons:

- **Graphics**
- **Text File**
- **Wait Mouse**
- **Mouse Interrupt**
- **Data Form**
- **Wait Key**
- **Keyboard Interrupt**

When you enter the Editor, you will see a message that reads *Display Object Editor*. To view the pull-down menus of available options, press and hold the right mouse button. Two menus will be displayed: the Project Menu and the Objects Menu. You use the Project Menu to do things like specify the background picture, save the objects, or exit the editor. Use the Objects Menu to add rectangles, polygons, lines, circles, ellipses, text, brushes and fields. These objects may also be moved, re-sized and changed with Objects Menu options. When you have completed your screen display, select Exit from the Project Menu.

The menu items are described in detail on the following pages.

The Project Menu



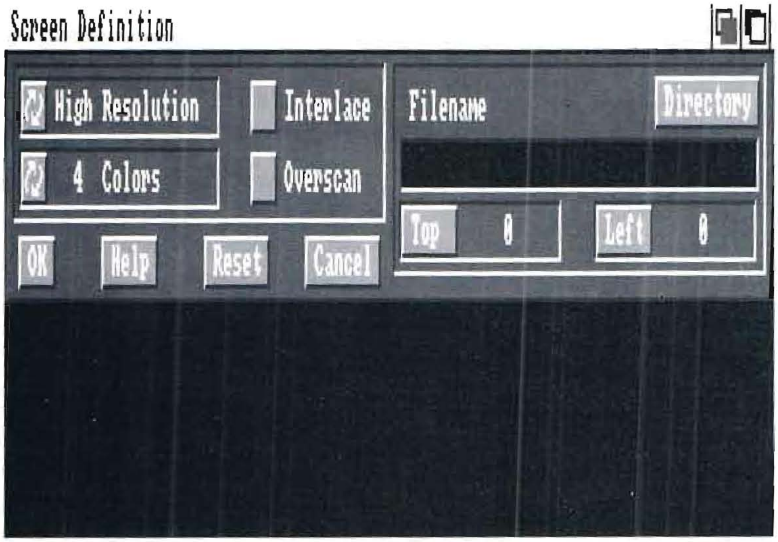
Screen

Refers to the background screen on which you place display objects. This has four submenu options: Definition, Palette, Videodisc and Clear. The Screen menu sets the background screen definition only during the editing process, not during runtime. In runtime the background screen definition is determined by preceding icons.

The Screen option merely helps you recreate the background of an application for the sole purpose of creating and positioning the display objects and hit boxes. The background pictures or videodisc frames you select with the screen editor are not automatically displayed in the AmigaVision application. You must use the AudioVisual icons to specify the screen background in your actual runtime application.

Definition

Allows you to specify the background picture and/or the screen resolution. When you choose the Definition option, the Screen Definition requester is presented. Here you may specify a picture file that is displayed temporarily as a background to help give visual cues for the design of display objects. This lets you correlate and position your hit boxes with the underlying graphic.



Screen Definition

High Resolution

4 Colors

Top 0

Left 0

The Screen Definition also lets you set the resolution you want without having to load a background picture at all. The table below shows all the resolutions that are available. As you can see, for each screen resolution (Low Res, Hi Res, Extra Halfbrite, and Hold and Modify) there are two possible options; i.e., interlaced and/or overscan. It is assumed that you are familiar with the screen options available for your Amiga.

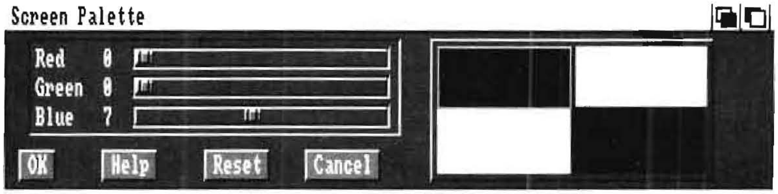
	Overscan		✓ Overscan	
	Interlace	✓ Interlace	Interlace	✓ Interlace
High Resolution 2 Colors 4 Colors 8 Colors 16 Colors	640 × 200	640 × 400	704 × 240	704 × 480
Low Resolution 2 Colors 4 Colors 8 Colors 16 Colors 32 Colors	320 × 200	320 × 400	352 × 240	352 × 480
Extra Halfbrite 64 Colors	320 × 200	320 × 400	352 × 240	352 × 480
Hold & Modify 4096 Colors	320 × 200	320 × 400	352 × 240	352 × 480

The table above shows the various screen dimensions (in pixels) for each screen resolution mode. For instance, if you select low-resolution screen with Interlace and Overscan, then the pixel resolution will be 352 in horizontal direction and 480 in the vertical direction. You also have a choice of between 2 and 32 colors for this resolution. In contrast, the Hold and Modify mode (HAM) has a fixed set of 4096 colors regardless of whether you choose Interlace or Overscan.

There is also an extended overscan feature for even larger size pictures. This is set from the defaults requester accessed from the main project requester.

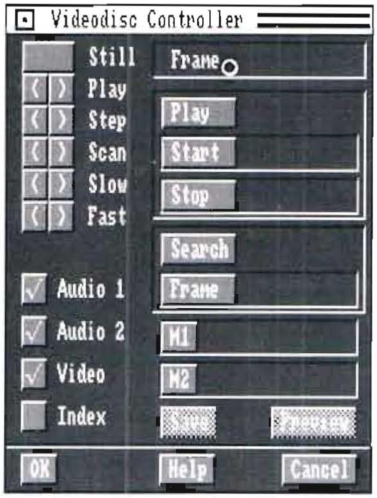
Palette

Allows you to modify the screen colors of a picture or animation. The Palette requester is presented showing all the colors in the current screen background. Each color is made up of a combination of Red, Green and Blue values. Each of these three colors has a range of 16 possible values from 0 to 15. A Slider gadget on the left side of the Modify Palette requester is provided for adjusting the Red, Green and Blue values. To adjust a palette color first you select a color from the palette of colors by clicking on it. The Slider gadget will show the three color values for that color. Then you can drag the three slider gadgets to modify that color.



Videodisc

Selects a frame on the videodisc to be used as a stationary background image. When you select this option, the Videodisc Controller is displayed. You use the Controller to search for the appropriate background frame of interest.



☑ Audio 1
☑ Audio 2
☑ Video
□ Index

Clear

Erases the current background without removing the hit boxes or other display objects that are on the screen.

Load

Allows you to load display objects and background screen information previously saved in a file. When you create display objects you have the option of saving them in a separate file (with Save and Save As), perhaps as a part of a library of files of standard objects. This file is independent of any application. The Load option allows you to load display objects and the background screen from these files. Thus objects you have created for one application can be reused for another. These files have a '.dob' extension.

Clear All

Removes all currently defined display objects and lets you start again. This option can be used to delete current objects if you are not satisfied with the design or to re-edit from the start without erasing the background screen definition. The Clear All option does not clear the background picture or video. To clear the background use the *Clear* option.

Preview

Displays the objects you have created on the screen over the specified background as they will actually appear to the user of the application. Any hit boxes defined will be active, and they will simulate the specified actions such as feedback sounds, texts and images. The Preview option is useful in testing the behavior of display objects and hit boxes after creating them. Once you enter the preview mode, it remains in effect until you click the right mouse button, which gets you back to the edit mode.

Redisplay

Redraws the currently defined display objects (refreshes the screen display).

Save

Saves the display objects and screen background specifications to a file. If a filename is not specified by either the *Load* or the *Save As* options prior to using the *Save* option, then a File requester is presented for you to specify a filename. If a filename is already specified, the Save option simply updates that file.

Save As

Specifies a filename for saving display objects and background screen information. Use this when saving a file for the first time or to change a filename.

Please note that the *Save* and *Save As* options are merely for saving the display objects and do not have anything to do with a specific AmigaVision application. These options allow you to build up a library of display objects and hit boxes for use in multiple applications. You can access the saved object library files using the Load option as described earlier.

Help

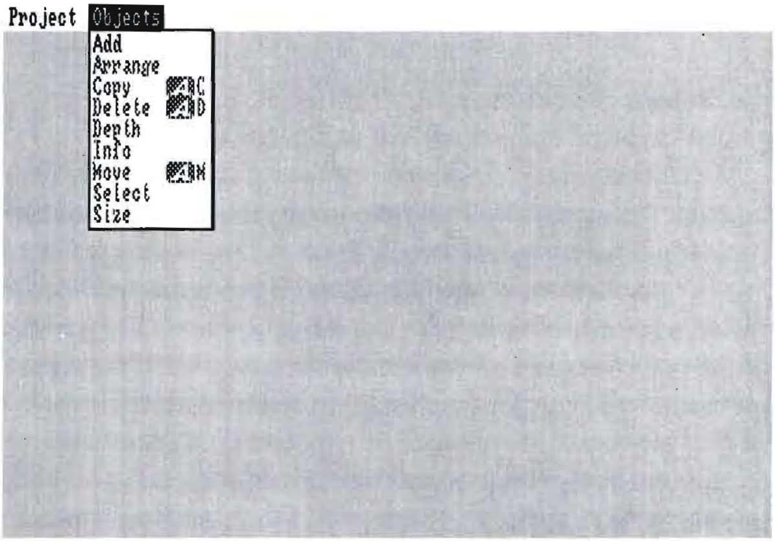
Gives you online help in using the Object Editor in general.

Exit

Allows you to exit the editor. If you entered the Object Editor from the pull-down menu of AmigaVision's flow editor and if you exit the Object Editor without saving (e.g., by using either the Save or the Save As options), then you will simply lose the display objects you created. If objects have not been saved for reuse, the editor will ask if you wish to do so at this time.

If, however, you entered the Object Editor through one of the icons such as Wait Mouse, the display objects and hit boxes that you create are *automatically saved* to the specific icon in the application when you exit the Object Editor using the Exit option. Thus you need not use the Save or Save As options unless you want to save the objects in a separate file for reuse later.

Objects Menu



Add

Allows addition of new objects or hit boxes over the background on the screen display. When you select the Add option using the right mouse button, you will see a submenu listing the object types available. These object types can be grouped into two categories: Geometric Objects and Special Objects. The first five, Rectangle, Polygon, Line, Circle and Ellipse, are geometric objects representing their respective shapes. The remaining four objects, Text, Brush, Input Field and Text Window, are special objects. We will address the geometric objects first.

Rectangle

Position the mouse pointer to mark the top left hand corner of the rectangle you want to create, press the left mouse button and drag the cursor until the rectangle is of the desired size, then release the button.

Polygon

Position the cursor at the starting point of the polygon. Click the left mouse button. Move the cursor to the next point and click again. Continue until the polygon is complete except for the very last line that links to the starting point. The last side of the polygon closes automatically when you click the right mouse button. Polygons are useful in creating arbitrarily shaped hit boxes. For instance, if you have an object that is irregular in shape (such as a picture of a dog or a cat), you can use polygons to create an object to suit the irregular shape.

Line

The line is only one pixel wide, and you cannot vary its thickness. Position the cursor at the starting point of the line, press the left mouse button and drag the cursor in the desired direction. When you are satisfied with the length of the line, release the mouse button. The line option is more useful for drawing lines on the screen for graphic display purposes, rather than for creating hit boxes.

Circle

Position the cursor to mark the center of the circle, press the left mouse button and drag the cursor until the circle is of the desired size. Release the mouse button.

Ellipse

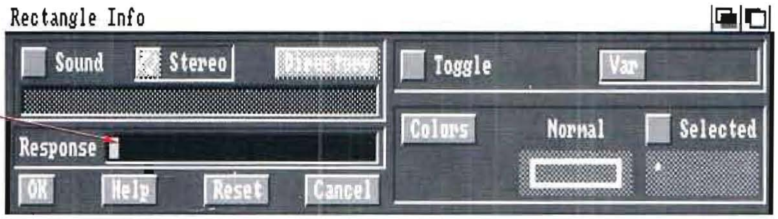
Position the cursor to mark the center of the ellipse. Press the left mouse button and drag the cursor until the ellipse is the desired shape and size. The semi-major axis of the ellipse must be either horizontal or vertical.

Please note that display objects created in one resolution will become distorted if you later change the screen resolution.

Defining Display Object Attributes

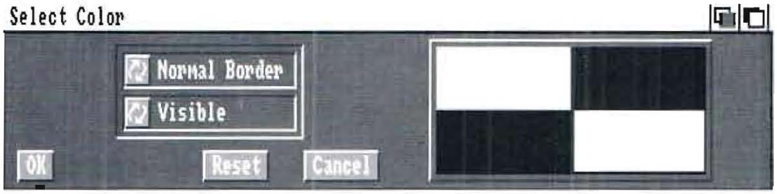
In each of the cases above, as soon as you create the object it will be in a "selected state," indicated by a moving border. You can also select any object by clicking on it once. Only one object may be selected at a time. Double-clicking on an object displays the requester for that object for you to define the attributes of the object.

The requesters for each of the five geometric objects described above are identical. In these requesters you can specify the normal border and the interior colors of the object, the border and interior colors for feedback effect, a digitized sound file for feedback and a response string if you want the object to be treated as a hit box.



type a response string here to create a hit box

To specify the color of the object, click in the Colors gadget at the bottom right section of the requester. You are presented with another requester called Select Color to specify colors for the border outline and the interior color of the object as it is displayed.



The top Multistate gadget on the left hand side of the *Select Color* requester is for selecting one of four items:

1. ***Normal Border color***
2. ***Normal Fill color***
3. ***Select Border color (for Feedback)***
4. ***Select Fill color (for Feedback)***

Select each of these four items and then specify whether you want a Visible color or a Transparent appearance by clicking on the second Multi-state gadget.

The *Visible* option selects one of the colors displayed on the right side of the requester by clicking on it. The selected color will be highlighted by a rectangular box around it. The *Transparent* option prevents you from specifying any color. After you have specified the colors for all the four states for Normal and Select colors, then you can click on the *OK* gadget to return to the object requester. The colors of the normal and feedback states are displayed at the bottom right corner of the Object requester.

You may select audio feedback for the object by clicking on the *Sound* gadget at the top left corner of the object requester. If you do not specify any sound filename, the default audio feedback is a short beep sound. If you want to specify a digitized sound file, then type it in the filename field below or use the *Directory* gadget to select a file. You can select whether the sound will be played back on the Left or Right speakers or as stereo by clicking on the multi-state gadget provided for it.

The *Toggle* gadget turns the toggling feature on or off. When it is on, and when a user clicks on the object, the normal image changes to the feedback colors. The feedback image remains until the user clicks on the object again, at which time it is restored to its normal state. If you set Toggle to off, the feedback colors appear only for the duration of a click.

The *Var* gadget links a variable to the object you have created. When you click on the *Var* gadget a list of previously defined variables in your application will appear. Select a variable to create the link. The variable you select must be of the Boolean type; i.e., it returns either a TRUE or a FALSE value. This means that if the object is selected by the user, the variable linked to it will have either a TRUE value or a FALSE value depending on the state of the toggle.

If the *Toggle* gadget is set to ON then the FALSE value of the Boolean variable will be associated with the normal state of the object and the TRUE with the feedback object.

For example, if you create a switch to turn a specific sound on and off, then you can use the Boolean variable to check the state of the switch. If the Toggle is set OFF, the Boolean variable will have a TRUE value as long as a user points to the object and holds the left mouse button down. When he/she releases the button, the variable will return to FALSE state. This feature can be used to simulate a gadget that works like a typical remote control unit for operating televisions and video cassette recorders.

Creating a Hit Box

Once you create an object you can turn it into a hit box by typing a string in the Response field. This response string gets linked to the object and turns it into a hit box. If you do not specify a response string, the display object will be treated as a visual object with no other function. The hit box, on the other hand, is very useful in finding out which of several boxes on the screen was clicked on by a user. Thus, the response string acts like a name for the hit box, and when a user clicks on it, the response string is passed on to an AmigaVision function called response() (see the Expression Editor section for standard functions). Thus, in your application flow you can use a Wait Mouse icon with a number of hit boxes representing a number of choices from a menu. When a user selects one of them, you can use conditional icons such as an If-Then icon or an If-Else icon to test the response() function for the response string indicating which of the menu choices were made. See the Control icon sections for examples of menus.

Note that when you enter the Object Editor through the Graphics icon (an Audio Visual icon), you cannot specify hit boxes. The Graphics icon is used merely to create displayable graphic shapes, and not active hit boxes.

Special Objects

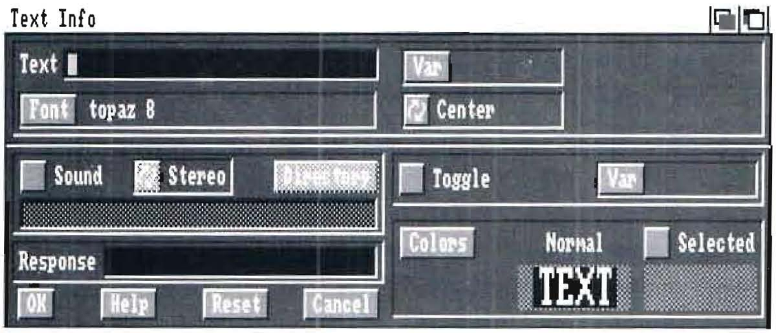
There are four special types of objects that you can choose from the Add option in the Object Editor.

1. ***Text/Variable***
2. ***Brush***
3. ***Input Field***
4. ***Text Window***

Text/Variable Objects

Text/Variable objects are similar to the geometric objects except that they are made of text strings. You can create text strings that fit on a single line (this depends on the resolution and the font size, with a possible maximum of 255 characters). Text may be placed anywhere on the screen. If the string is longer than a single line on the screen, the text does not wrap around to the next line.

When you select *Text* from the *Add* submenu, and click on a location on the screen marking the starting point of the text string, you are provided with a small rectangle to size the length of the text string. You size the string length by dragging the mouse. You can then double-click on the rectangle to access the Text requester. Normal, Feedback, Toggle, Colors, Sound Feedback and Response for the Text requester work the same way as for the other object requesters described earlier. Just as with graphics objects, the text object can also be defined as a hit box by specifying a Response string.



Text Info

Font topaz 8

Center

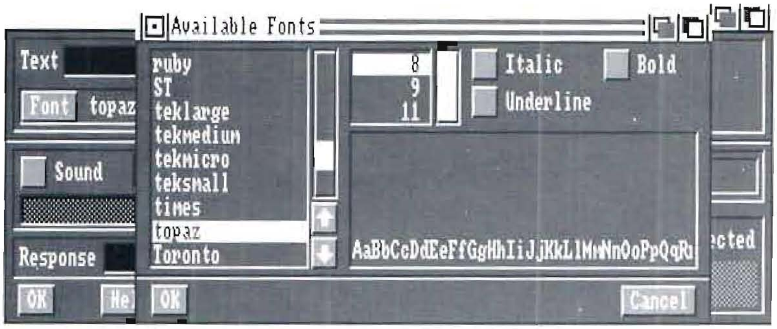
You can create composite objects by creating a geometric shape to denote say, a hit box, and a text string to give it a name such as "Help". In such cases you have to define the same response strings for both of these objects in order for them to function as a single composite object.

In the Text Info requester, there is one additional feature called Variable. This feature lets you print on the screen the value of any variable in your application.

To display the contents of a variable click on the *Var* gadget at the top right to select a variable from a list whose value you want to display. You can type the string to be displayed at the text string field at the top left part of the requester after clicking on it. The variable value will be appended to the end of the text string when displayed under presentation of the application.

Click on the Multi-state gadget to choose whether you want the text string and/or the variable value printed *Center, Left* or *Right* justified within the text object rectangle.

Click on the *Font* gadget to open the Available Fonts requester. Click on the name of the font you wish to use and select the font size. You may also select Italic, Boldface or Underline by clicking on these gadgets. The window at the bottom right of the Font requester displays the appearance of the text in the font, size, and features you have selected. Click on *OK* to save these specifications and exit the Available Fonts requester.

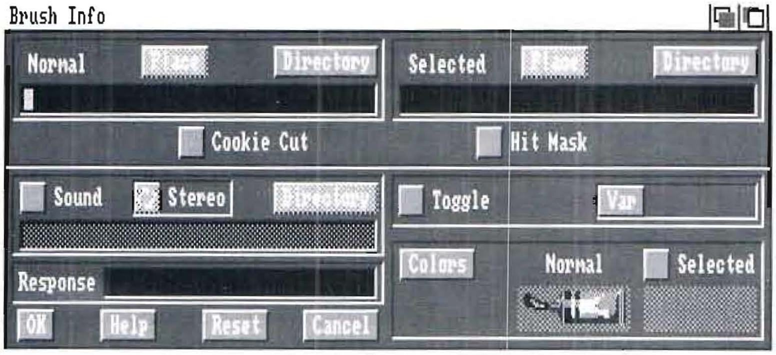


Once you are satisfied with the attributes of the text object, you can exit the Text Info requester by clicking on the *OK* gadget. The text will be placed at the location of the text object rectangle. You can resize the rectangle to suit the length of the text string. If you have specified a variable to be displayed, the name of the variable will be displayed at the end of the string. During presentation of the application, the current value of the variable will be displayed along with the text string object.

Brush Objects

Allow you to place on the screen an image which has been previously saved in an IFF format (e.g., under DeluxePaint III you can choose the brush option for saving). A brush can be defined either as an active hit box-type object, or as a non-responsive displayable graphic shape, just like text and geometric shapes. The brush option gives you the power to create customized pictorial hit boxes that are both attractive and descriptive.

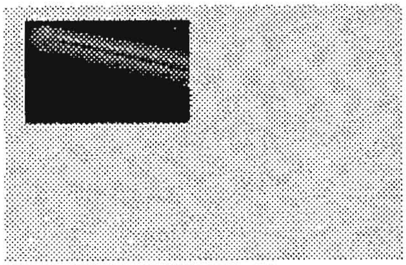
When you select the *Brush* option from the pull-down menu, use the cursor to mark the top left hand corner of the rectangle where you want the brush positioned. Press the left mouse button and drag the cursor to the desired rectangular size. When you are satisfied, release the button. Double-click inside the rectangle to access the Brush requester.

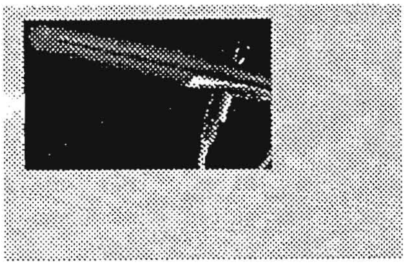


To choose the brush you want to display, click in the field below where it says Normal and either type in a brush filename or use the *Directory* gadget to select a brush filename. Click on the Place gadget and you will exit the Brush requester. You will see that the brush image will appear inside the rectangle you had specified for the brush flush with the top left hand corner.

Now if the rectangle you had defined were smaller than the size of the brush, you could click on the brush and move it around and place it on the rectangle. While placing the brush, you can use the rectangle as a frame through which you show only a part of the brush image. For instance, if you have a brush image of a human body and you wish to use only the head as a brush object, then you use the *Place* object to show just the head through the brush rectangle. The part of the picture outside of the rectangle will not be seen. You can use Size option from the pull-down menu and resize the brush rectangle to include a larger area of the picture as a brush object.

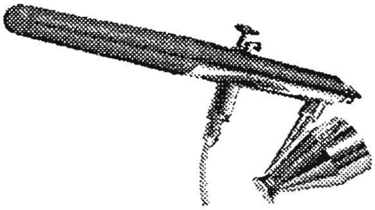
If the brush rectangle is larger than the size of the image then, by using the Place gadget you can locate the brush. Once it is placed, you will be returned again to the Brush requester.



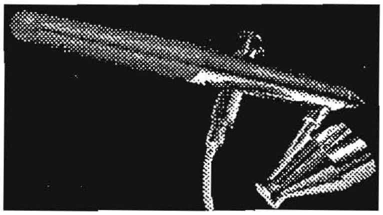


If you wish to display a brush as feedback, follow the same procedures as choosing the Normal brush. The gadgets corresponding to the Feedback brush are provided under Select. By using both the Normal and Feedback brushes you can create special effects.

There are two on/off gadgets called *Cookie Cut* and *Hit Mask*. Normally brush objects are displayed with their background rectangles. If you choose *Cookie Cut* by setting it ON the brush will be displayed without a rectangle in the background.



☑ Cookie Cut

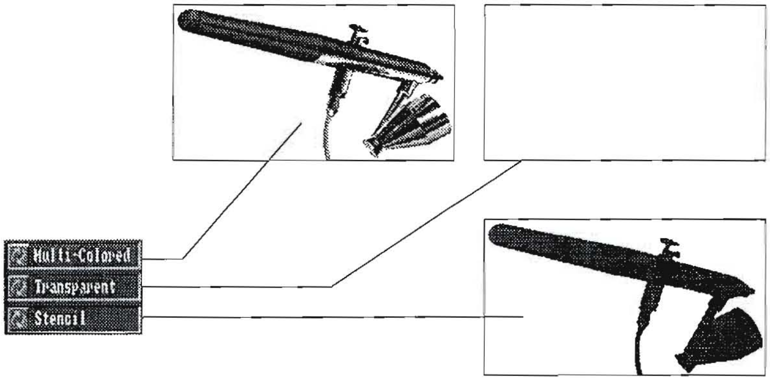


□ Cookie Cut

Normally, when you define a brush object as a hit box using response strings, the entire background rectangle is treated as a valid region for the hit box. Frequently the shape of the brush within the rectangle may be arbitrary. In such cases you might want to restrict the hit sensitive region to lie within the boundary of the brush. If you set *Hit Mask* ON, your intended user must click directly within the boundary of the brush to activate the hit box. When *Hit Mask* is set OFF, which is the default, the user may click anywhere within the rectangular background area to activate the hit box.

If you click on the *Color* gadget of the brush requester you will be presented a color selection requester. For the normal and the feedback brushes you have the option of displaying the brush as:

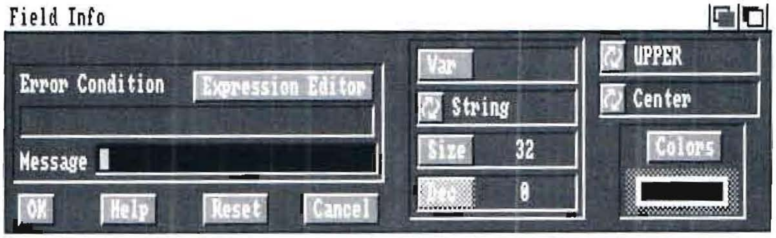
- **Transparent.** Creates a see-through brush object.
- **Multi-Colored.** Displays the brush in its original form in the palette of the background screen.
- **Stencil.** Created when you choose a solid color for the brush. This is useful to create an effect in which you hide the brush until a stencil is clicked on and reveal the brush in full colors as a feedback.



The rest of the Brush requester is similar to other object requesters discussed earlier. The Response string, Toggle and Sound feedback are identical in operation to the other cases.

Input Field

You can use *Input Field* objects to allow users to enter their name, security number, password, etc. and assign these data to variables in your application. The field is used for both entering and displaying the current value of the variable. When you select the Input Field option from the Add submenu, the cursor will appear as a crosshair. Use the crosshair to locate the top left corner of the Input Field rectangle and then drag the mouse to create an input field of desired length, then release the button. The width of the field is restricted according to the type of variable linked to that field. For numeric values you can have up to 19 digits, and for string variables you can have up to 255 characters. But in practice the string field is limited to one line of text. Double-click on the field to enter the Field Info requester.



String

Size 32

Dec 0

UPPER

Center

Note that input fields can only be created if you entered the Object Editor through the Data Form icon or if you select the Object Editor from the pull-down menu. If you create input fields after entering from the pull-down menu, they are not saved to any specific location in the project, but may be saved as a separate file for re-use.

To access the variable list for selecting the variable to be linked to the field, click on the *Var* gadget on the right hand side of the requester. You use the Multistate gadget to select the specific field type: String, Numeric, Boolean or Date. The type should correspond to the type of the variable you have specified.

To define the precise length of the field as a number of characters click on the *Size* gadget. A numeric pad will open, allowing you to enter the number. If the field is numeric, you can further specify the number of decimal places by clicking on the *Dec* gadget.

For string fields, click on the Multistate gadget to choose *UPPER, lower* or *MiXeD* case. The input will be transformed into the specified case. If you specify upper case and the user types lower case, the characters will become upper case when he presses the Enter key. If you wish to have the field insensitive to cases, then use the Mixed mode.

Click on the Multistate gadget below the case gadget to select the justification of the input data: *Center, Left* or *Right.*

Click on *Color* gadget to select the color for the field border.

The *Error Condition* field allows you to specify a conditional expression to check the validity of the data entered into the field. Click on the *Expression Editor* gadget to access the Expression Editor and specify the appropriate condition. The condition for the input field exit that you specify may be an expression that validates the data for accuracy, syntax or consistency. When the user entering data in a field presses the Enter key on the keyboard, it is assumed that the data entry has been completed. At this point AmigaVision compares the user's input to whatever condition exists in the Exit condition field. A check is made to see if the specified conditional expression is satisfied; i.e., returns TRUE. If not, the cursor stays in the field, and an error message can be displayed. When no condition is specified, the field will be exited after the user presses the Enter key.

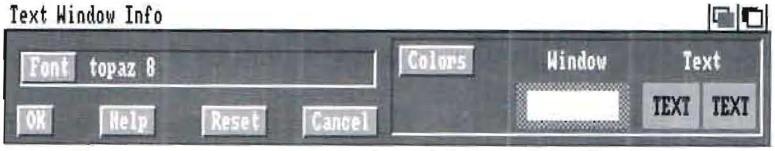
You can specify your own error message right below the conditional expression. Type the message in the Error field. This error message will be displayed if the input is invalid.

Text Window

Text Window allows you to create a rectangular display area (a window) on your screen for viewing an ASCII text file. You can also add functions through hit boxes to control the display of text such as page up, page down, line up, line down and exit so that a user can browse a text file.

Note that the Text Window option is only available when you enter the Object Editor through the Text File icon or from the pull-down menu. If you create text windows after entering from the pull-down menu, they are not saved to any specific location in the project, but may be saved for re-use.

Choose *Text Window* from the *Add* submenu. A crosshair cursor will appear. Use the cursor at the upper left corner where you want the text window to display. Press the left mouse button and drag the rectangle to the desired size. The letters ABC will appear in the first two lines of the top left corner of the rectangle. The border will be highlighted to indicate the selected state for this object. Double-click inside the rectangle to access the Text Window Info requester.


Text Window Info

Font topaz 8

Click in the *Color* gadget to choose the colors of the window Border, the Window Background, the Normal Text, and the Highlighted Text.

There are several commands you may imbed within your text files to control font styles and formatting. Text commands are signalled by using the | (vertical bar) character followed by a number code. For instance:

This is some |Bbold text | B, and |Iitalic | I text

appears in the text window as:

This is some **bold** and *italic* text

Notice that the command to initiate bold text, | **B**, is placed directly before the section to be made bold. The commands for font styles (highlight, italics, bold and underline), act as switches, and stay on until turned off with the same command. To print the | character itself in a document, use | |. These style commands may also be nested, so

Here is some |B | Ibold AND italic |I| B text

appears in the text window as:

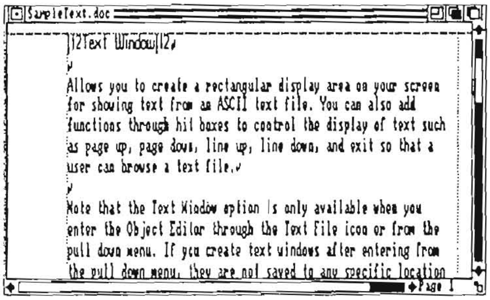
Here is some ***bold AND italic*** text

Font Style Commands

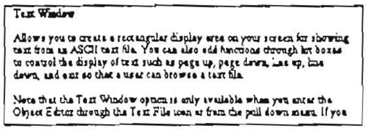
|H Highlight
|I Italics
|B Bold
|U Underline

Most likely you will be using a word processor to compose text files. Then you save the files in ASCII format. In a word processor, carriage returns (CR's) are typed only to start paragraphs, since most word processors use automatic word wrap. The default formatting method used in AmigaVision text file boxes is to word-wrap sentences to fit within the horizontal size of your box, and to start paragraphs when a

Carriage Return (CR) is detected. If you have switched to another formatting method, using |**FD** will switch back to the default word-wrap method.



```
SampleText.doc
|12Text Window|12↵
↵
Allows you to create a rectangular display area on your screen
for showing text from an ASCII text file. You can also add
functions through hit boxes to control the display of text such
as page up, page down, line up, line down, and exit so that a
user can browse a text file.↵
↵
Note that the Text Window option is only available when you
enter the Object Editor through the Text File icon or from the
pull down menu. If you create text windows after entering from
the pull down menu, they are not saved to any specific location
```

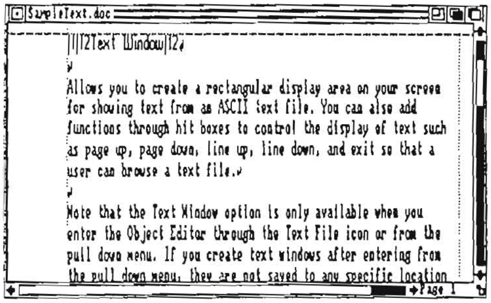


Text Window

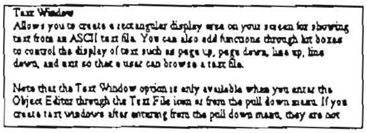
Allows you to create a rectangular display area on your screen for showing text from an ASCII text file. You can also add functions through hit boxes to control the display of text such as page up, page down, line up, line down, and exit so that a user can browse a text file.

Note that the Text Window option is only available when you enter the Object Editor through the Text File icon or from the pull down menu. If you

Some editors, however, do not use word wrap, since they are used primarily for programming. Because of this, the size of your text editing window may differ from the size of your text output window in AmigaVision. To avoid unsightly gaps in the text, using |**FW** before a section of text will cause AmigaVision to ignore any CR's it detects unless there are two in a row (which you would use in the text editor to denote a paragraph break).



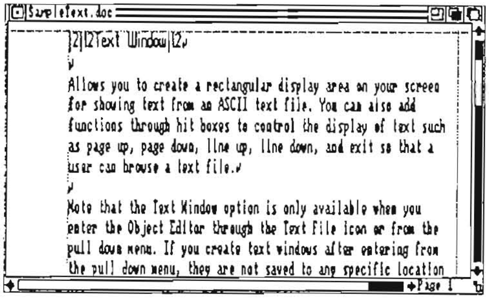
```
SampleText.doc
|||12Text Window|12↵
↵
Allows you to create a rectangular display area on your screen
for showing text from an ASCII text file. You can also add
functions through hit boxes to control the display of text such
as page up, page down, line up, line down, and exit so that a
user can browse a text file.↵
↵
Note that the Text Window option is only available when you
enter the Object Editor through the Text File icon or from the
pull down menu. If you create text windows after entering from
the pull down menu, they are not saved to any specific location
```



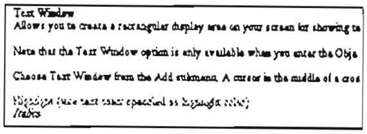
Text Window
Allows you to create a rectangular display area on your screen for showing text from an ASCII text file. You can also add functions through hit boxes to control the display of text such as page up, page down, line up, line down, and exit so that a user can browse a text file.

Note that the Text Window option is only available when you enter the Object Editor through the Text File icon or from the pull down menu. If you create text windows after entering from the pull down menu, they are not

Finally, there may be occasions when you do *not* want AmigaVision to word wrap your text to fit in the box. Putting |**FN** before a section of text will allow only as many characters as will fit in one box-width to be displayed, until a carriage return is detected.



```
Sampletext.doc
]2|I2Text Window|I2↵
↵
Allows you to create a rectangular display area on your screen
for showing text from an ASCII text file. You can also add
functions through hit boxes to control the display of text such
as page up, page down, line up, line down, and exit so that a
user can browse a text file.↵
↵
Note that the Text Window option is only available when you
enter the Object Editor through the Text File icon or from the
pull down menu. If you create text windows after entering from
the pull down menu, they are not saved to any specific location
Page 1
```



Text Window
Allows you to create a rectangular display area on your screen for showing te

Note that the Text Window option is only available when you enter the Obje

Choose Text Window from the Add submenu. A cursor in the middle of a cros

Highlight (uses text color specified as highlight color)
Italics

Text Formatting Commands

|**FD** Normal word wrap inside text box size, one CR per paragraph break.

|**FW** Normal word wrap inside text box size two CR's per paragraph break.

|**FN** No word wrap inside text box.

To choose the font you want, click on the *Font* gadget. This opens the Fonts requester. Select the font, size and style from this requester.

You can specify only one text window on a single screen. You can add user control features by defining hit boxes of any shape or a brush. For the hit boxes you must use the following response strings in the hit box requesters:

- "page down" for user to move to the next page of the text file.
- "page up" for the previous page.
- "line down" for the next line (like a line scroll).
- "line up" for the previous line.
- "quit" for closing and exiting the text window.

The Text Window is very useful in displaying large text files and allowing the user to browse. See the Text File icon section, which is a part of the chapter on Audio Visual icons, for a description of an example of the text window.

Arrange

Is for reordering objects. The order of objects refers to the sequence number assigned to each object on the screen in relation to others. It is as if the objects are placed on a stack on top of the screen. The object you want on top can be placed in the foreground. This feature is particularly useful in ordering the sequence of operation of input fields. For instance, if you want the user to enter data in a specific sequence, then you must sequence the fields correspondingly using the Arrange option.

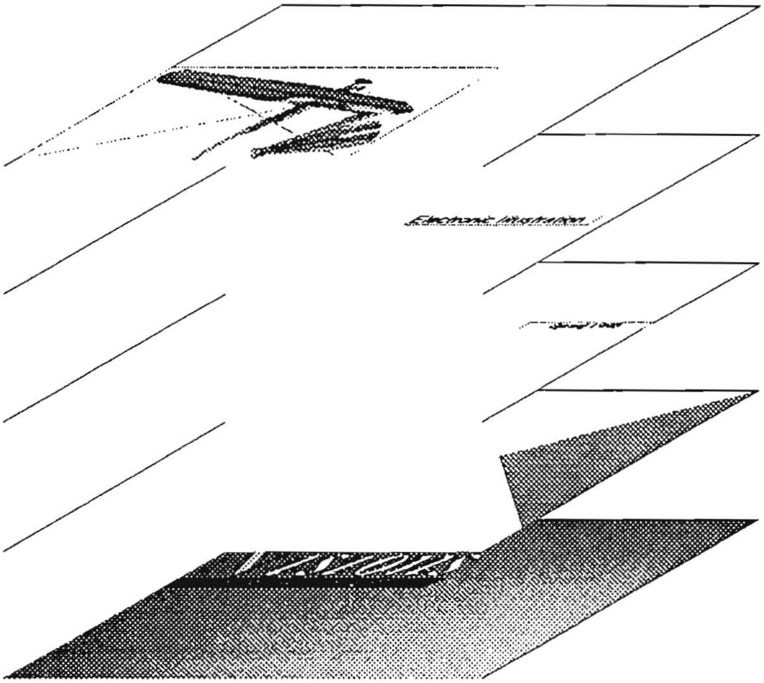
To sequence objects after you select this option, click on them in the desired order. The first object clicked will be the first in the sequence. The next object you click on will become the second object, and so on. This continues until you click the right mouse button.

Copy

Duplicates the currently selected object. First click on the object you want copied, then choose Copy from the menu. The cursor will change to a copy cursor. Press the left mouse button, drag the new object to the appropriate position, and release the mouse button. The object will now appear in both places. Click outside the object when you are done, or double-click in the object if you wish to make changes in the requester. This option is useful in creating multiple hit boxes of identical shape.

Delete

Erases the currently selected object. First select the object with a mouse click, and then choose Delete in the menu. The selected object will be erased.



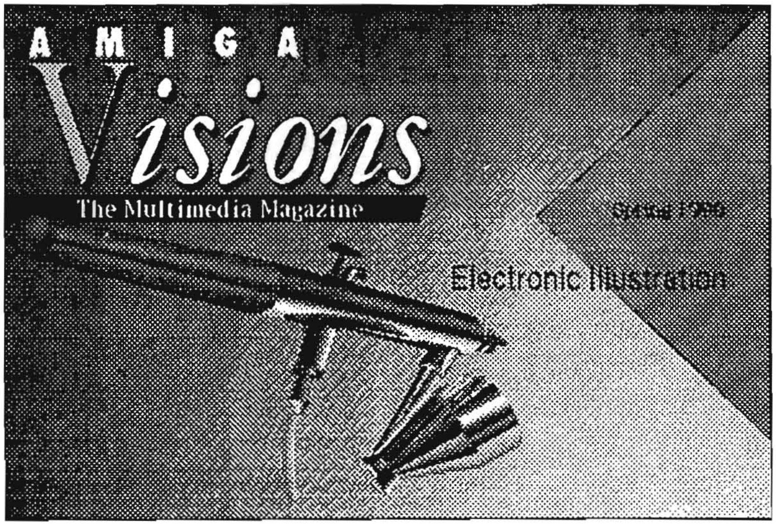
Airbrush image

"Electronic Illustration" text object

"Spring 1990" text object

Triangle

Background image with magazine name



The illustration to the left shows how objects on separate layers in the Object Editor are used to build up the cover of a fictitious disk magazine. Elements that change each month, like the main topic, can be added as text objects, and illustrations can be added as brushes over the background image.

Depth

Determines which objects are drawn on top of others. Consider all the objects to be in a stack. Depth controls the position of the current object in the stack. The choices are:

Front Moves the selected object to the top position.

Raise Moves the selected object up one position.

Lower Moves the selected object down one position.

Back Moves the selected object to the bottom position.

Click on the object, then select Depth from the Objects Menu. Select the desired position. The object will move to its new position.

Please note that since you can have overlapping hit boxes or display objects, you need the Arrange and Depth options to define which objects are in front of the others. The object in front gets recognized first and the rest in the order of the sequence.

Info

Displays the requester for the currently selected object. This can be used to define or change the definition of an object. Click once on the object and choose Info from the Objects Menu. The requester will appear for that object. You may also enter the requester by double-clicking on the chosen object.

Move

First click on the object to be moved, then choose Move from the Objects Menu. The cursor appears as an arrow. Point to the object you wish to move and press the left mouse button. Drag the object to the desired position and release the mouse button.

Select

Chooses an object to be edited. It is useful when you need to edit an object which is under another one. Choose Select from the Objects Menu. Move the cursor to the object of your choice. The object selected will have a flickering border indicating a selected state. To edit, double-click inside the object and you will access that object's requester. There are three submenu choices:

1. ***First*** Selects the top object.
2. ***Next*** Selects the next object down from the currently selected object.
3. ***Previous*** Selects the next object up from the currently selected object.

You may also simply select an object by clicking on it.

Size

First click on the object. Then, choose Size in the menu. The cursor appears as a box with two arrows. Position the cursor on the object to be changed, press the left mouse button and drag the cursor to adjust the object's size larger or smaller. When you finish, release the button. Polygons and brush images may not be re-sized.

Videodisc Controller

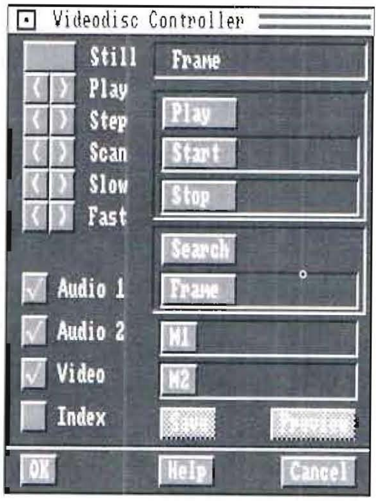
The Videodisc Controller can be accessed through the requester of the Video icon to define video sequences or display selected frames. If you just wish to browse a videodisc, then you can access the Videodisc Controller from the pull-down Tools Menu.

The Videodisc Controller allows you to view video, save frame numbers, and perform many other browsing functions. The frame numbers may then be saved and used with the video icon to include video in a course or presentation.

Before operating the Videodisc Controller, please make sure that you have a videodisc player attached to the computer with proper cables and a genlock to display video on the monitor. Also, make sure that the proper baud rates are set on the player, and that it contains a usable laser videodisc. In addition, make sure you have set up the proper settings from the Video Setup Project Menu, including the model of videodisc player and baud rate.

There are two types of videodiscs. One is in CAV (constant angular velocity) format, which allows random access by frame numbers. The other is CLV (constant linear velocity) format, which allows access by time. *AmigaVision's Videodisc Controller is principally designed for use with the CAV format.*

A choice of video commands is shown in the upper left of the window. Click on a command gadget to use the command. Notice there are two arrows to the left of the command name. These are used to specify forward or reverse. For instance, if you click on the left arrow next to the *Fast* gadget, the video will play fast reverse.



☑ Audio 1
☑ Audio 2
☑ Video
□ Index

Still pauses the video. *Play* shows the video at normal speed in forward or reverse. *Step* advances one frame at a time. To move forward or reverse at a very fast pace, point to one of the *Scan* arrows and press the left mouse button. Hold the mouse button down while scanning and release the button to stop. *Slow* shows the video in slow motion. *Fast* shows the video in fast motion.

Click on the *Audio 1* and *Audio 2* gadgets to change the status of the audio channels. A check mark indicates that the audio is in the ON state. The Audio switches function only in the forward play motion of the videodisc.

The *Video* and *Index* gadgets are used in a similar fashion, to turn the video and frame number index on or off. A check mark indicates the ON state. *Index* refers to the video frame numbers typically shown in the top area of the videodisc display. The *Video* on or off option is available only for certain video players since some players do not have this feature. For those players without this feature, the Video status gadget has no effect on the display of video.

You may also store two separate frame numbers temporarily in memory. To do so, click on either the *M1* or *M2* gadget. This will enter the number of the current frame into that memory area. You may use the frame numbers stored in this memory area in specifying the frame numbers for the Play and Search operations.

In the upper right section of the controller, *Frame* displays the current frame whenever the video is stopped or paused. *Play* will show the segment of video between the two specified frames and then stop. *Search* will advance the video to the frame specified.

Clicking on the *Start* or *Stop* gadgets in the Play area or *Frame* in the Search area displays the Frame Number requester. *Enter Frame* opens the Specify Value requester. *Current Frame* enters the current frame. If you choose one of the memory choices, the frame number saved in it will be entered.

The *Save* and *Preview* options are only available if you entered the Videodisc Controller from the **Video** icon requester, and they are disabled when you access the controller from the pull-down menu. To save the current settings and remain in the controller, click on *Play* or *Search* and then *Save*. This saves the state of the switches: *Video On/Off, Index On/Off, Audio 1 On/Off, Audio 2 On/Off* and the last executed command. To view the video as it will be seen in the course, click on *Preview.* When you are finished using the controller, click on *OK* to save the current settings and exit to the Video Requestor. Choose *Cancel* to exit without saving any changes.

While using the Videodisc Controller, you may wish to adjust its size. This will allow you to see more of the video display area. To make the controller smaller, click the right mouse button anywhere inside the controller. When you do this, only the commands at the top left side of the controller will be visible. When it is small, click the right button in the controller again to return to normal size. To make the controller disappear, click the right mouse button anywhere outside the controller. Click it again to make the controller reappear.

Database Editor

With the Database Editor you create and manipulate databases for use with projects. The Database Editor allows you to create a database in a standard database format; add, update and delete data records; as well as delete full database files.

You use the Database Editor from the pull-down menu only for database creation and data editing. You may access the databases you create and perform database operations from within your applications by using the database icons. The database operations include: Select, Read/Write, and Delete.

About Databases

Before explaining how the Database Editor works, we first must define some terms.

A **Database** is a collection of information which is organized to serve specific purposes. An example of a database would be a collection of personnel information for all the employees of a company.

A **Record** is a single occurrence of an item in the collection. An employee database would include one record for each employee. That record would contain all the information about the employee.

A **Field** is a category of information which is identified by a name and other attributes. For example, the employee database would include fields for employee ID number, last name, first name, middle initial, sex, salary, and so on. Each of these fields can be of a fixed length and can contain only a specified type of data (e.g., alphabetic or numeric, etc.).

You may also think of the database as a table of information. Using the table analogy, a record is equivalent to a row in the table, and a field is equivalent to a column.

A **Key** is a special type of field. Keys are used to order records and to facilitate searching in the database. When you use a telephone directory, you take advantage of the alphabetical organization and use a last name as a key to quickly locate a record. Similarly, when you designate fields in the database as keys, you will be able to arrange the records in order of the key field and to access particular records rapidly.

In the employee database, for example, you might want to arrange the records by employee ID number and also by employee last name. To do so, you would designate these fields as *keys.* You can also define compound keys, such as a key which is the last name field + the first name field. In this way, if you have more than one "Smith" in the database, you will be able to find the specific one named "Donald" for whom you are searching.

You must create a database prior to using the database icons — Select, Read/Write, or Delete — in your application flow. Please follow these steps in creating your database:

1. ***Use the Database Editor to define the record structure using data fields.*** This requires careful planning, since

in AmigaVision you cannot alter the record structure of a database once you have created it. For instance, you may want to create a database containing the name, address and phone number of the end user. Then each of the items — the last name, middle initial, first name, house number, street name, city, country, zip code and phone number — becomes a data field. If you anticipate adding a new field for age or sex later, you must either plan ahead or create a new database later.

The database is stored in a dBaseIII® compatible file format with a '.dBF' file extension. This allows you to use other commercially available database software such as Superbase, which supports dBaseIII-compatible files, to read AmigaVision data files for extensive analysis or modification. In AmigaVision you can also read dBaseIII-compatible files created by other database software, such as dBase, Paradox® and other dBase-compatible programs if the data is on an Amiga formatted disk.

2. ***Be sure to denote one or more of the data fields as keys for selecting a specific record.*** In the above example you may want to get to any record by specifying the last name and zip code. Then these two fields in your record become the keys for searching. AmigaVision allows you to change the keys even if you have specified them already. The keys are used to create what is known as an index file. On your disk associated with each .dBF file there is an .ndx file which stores the index of records based on the keys you have specified. When you change the keys, the old index file will be erased and replaced by the new one.

When you read dBaseIII-compatible files created by other software products such as Superbase, the keys specified in those products will not be transferred to AmigaVision. You must use AmigaVision's Database Editor and specify keys for creating the index (.ndx) files.

Field Types and Sizes

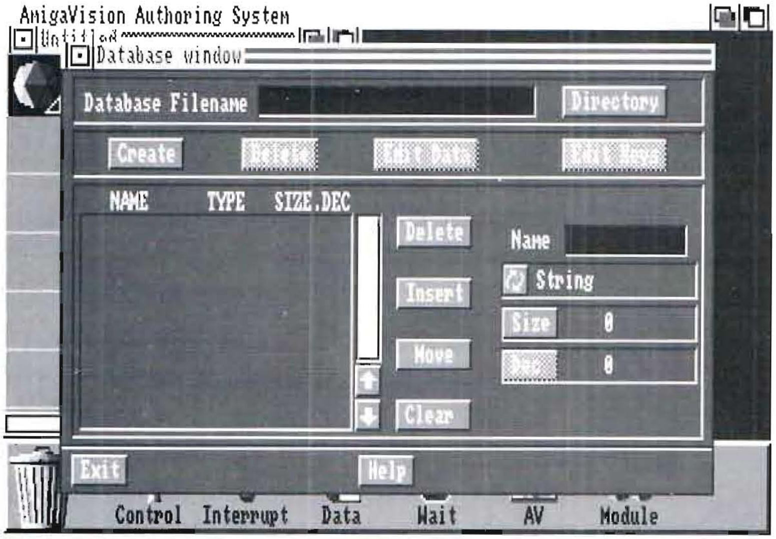
The AmigaVision Database Editor supports the following field types, which are shown along with their maximum allowable sizes:

- **String.** 255 characters maximum. String fields are used to store alphanumeric information like names, addresses, etc.
- **Numeric.** 19 digits maximum, including the decimal point. Numeric fields may contain integer or floating point numbers. If you select numeric, you may also specify the number of decimal places. The size of numeric fields includes the decimal and the number of places to the right of it.
- **Boolean.** Fixed width of one character. Boolean fields can only be **T** or **Y** (for TRUE or YES) or **F** or **N** (for FALSE or NO) and are assigned a default width of one character. These fields are like flags that you can set which allow you to search for records in which the condition is TRUE or FALSE. In the employee database, you might create a boolean field called "Eligible" which would be TRUE if the employee were eligible for a retirement plan and FALSE otherwise.
- **Date.** Fixed width of 10 characters. Date fields have the format MM/DD/YYYY, which is a string 10 characters long, (e.g., 12/10/1990.)

The database records that you create using the Database Editor may contain up to 128 fields. There is also a limit of 4000 characters per record.

Using the Database Editor

To enter the Database Editor, you select Database from the Tools Menu. The Database Window will then open.



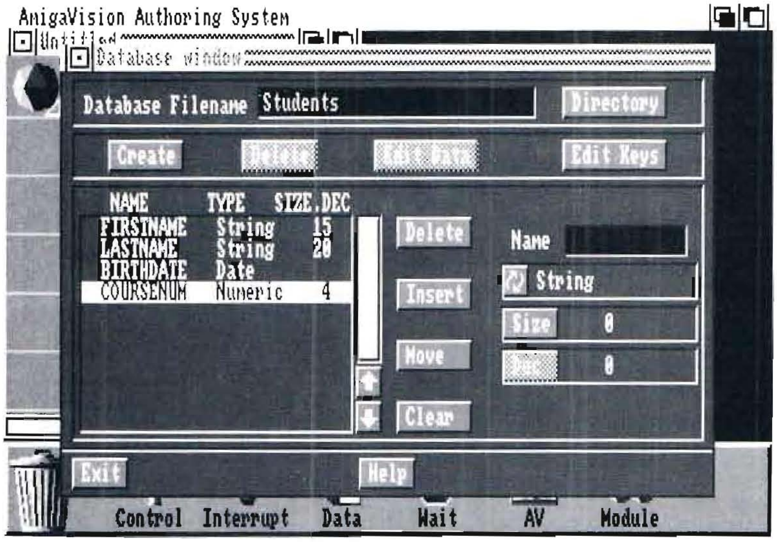
In the Database Filename field, you specify the name of your file. You may either type the filename into the field or click on the **Directory** gadget to select a file from the File requester. After you have specified a filename, press the Enter key.

If the file that you specify does not exist, you will be able to specify a record structure and choose the Create option, which creates a new database file. Note that once created, the record structure cannot be edited within AmigaVision. It can be edited, however, in other stand alone database software packages.

Step-by-Step Database Creation Example

Let's create a new database to keep track of students, their birth dates, the course they are taking, and whether or not they have paid their tuition.

1. *Type the name of the field in the Name field.* First we will call it "FIRSTNAME" by typing it in the field.
2. *You need to specify the length of each string. Click on the Size gadget and enter the number 15.*
3. *Insert the new field by clicking on the Insert gadget.*
4. *Define "LASTNAME" as you did above but change the Size to 20.*
5. *"BIRTHDATE" will track each student's birthdate. Enter it in the Name field.*
6. *BIRTHDATE will be a date, so click the Multistate gadget until Date appears. Dates are a fixed length of 10 characters, so just click on the Insert gadget.*
7. *Now define a number to represent the course the student is taking. Type "COURSENUM" in the Name field.*



Database Filename Students

NAME	TYPE	SIZE.DEC
FIRSTNAME	String	15
LASTNAME	String	20
BIRTHDATE	Date	
COURSENUM	Numeric	4

Name

String

Size 0

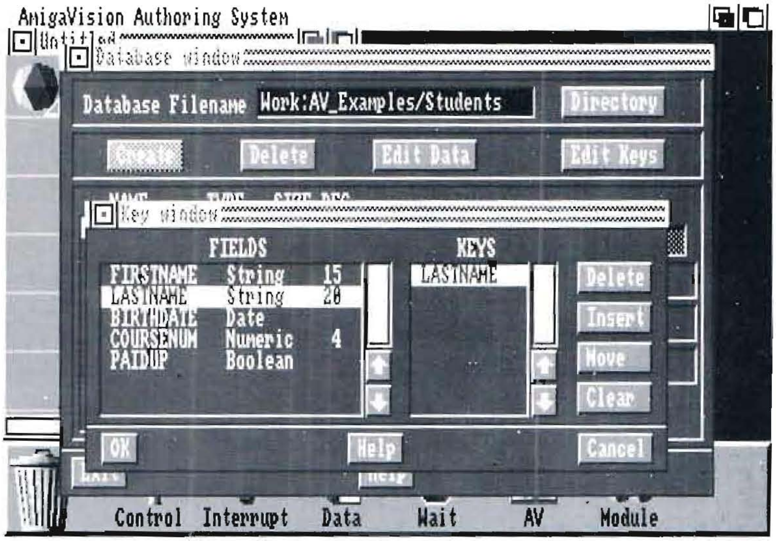
Dec 0

8. *Click the Multistate gadget until Numeric appears. Now define the Size of 4. Click on the Insert gadget.*

9. *Finally define a field to track whether tuition has been paid or not. Create "PAIDUP" by typing it in the Name field and setting the type to Boolean.* Boolean is another variable type that doesn't need a length, as it is always a single character representing either T for true or F for false. *Click on the Insert gadget.*

10. *Enter a filename for this example by either clicking on the Directory gadget and setting the path or by just typing a filename in the field.*

11. *Create the database on disk by clicking on the Create gadget.* A new database is now ready to accept data.



Database Filename Work:AV_Examples/Students

FIELDS

```
FIRSTNAME   String    15
LASTNAME    String    20
BIRTHDATE   Date
COURSENUM   Numeric    4
PAIDUP      Boolean
```

KEYS

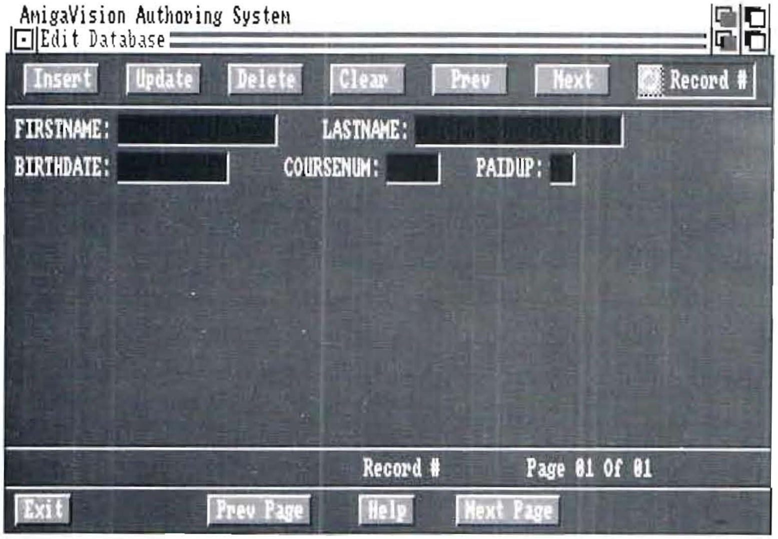
```
LASTNAME
```

12. *It is easiest to track these students by last name, so let's define LASTNAME as the key field. Click on the EDIT KEYS gadget and a new requester appears with a list of all your fields.*

13. *Select LASTNAME and click on Insert, and you'll notice LASTNAME now also appears in the KEYS list. Click on OK to save the key.* Your database structure is finalized and it is now ready for some data!

Editing Data

In the Database Editor you may edit (i.e., add, modify or delete) records in the database. To do so, type the name of the file in the Database Filename field, and click on the Edit Data gadget. The Edit Database Window opens.



Let's add some student records to our new database by clicking on Edit Data and bringing up the Edit Database Window.

1. ***Define the first student by typing in the data. Type your first name and press Enter.*** You will notice that the cursor jumps to the next field, where you can type your last name. Since the last name is a key field, it will be highlighted by a > character to differentiate it from the other non-key fields. Type in your last name in this field.

2. ***Type in your birth date.*** If your birth date is March 15, 1966, type "3/15/66", and AmigaVision automatically changes it in the field to "03/15/1966."

3. ***Type in a course number between 0 and 9999, since you defined this field as a numeric integer with a maximum of four digits.***
4. ***Finally answer the PAIDUP field with a "T" (or "Y") for TRUE or a "F" (or "N") for FALSE.***
5. ***Once you are satisfied with the data values you have entered, click on the Insert gadget at the top left to enter the record into your database.*** After inserting the current record, you can enter the next record and so on until you have entered all the data. After each insertion, a message at the bottom left part of the requester will provide a message confirming your record insertion and the input fields will be cleared to make them ready for the next record.
6. ***If you have entered the wrong information and want to clear all the fields, then click on the Clear gadget.***
7. ***If you want to update a previously inserted record, you can select that record using the Prev gadget, edit the data, and then click on the Update gadget.***

Selecting a Record

There are two methods by which you can select a record: **By Record Number** or **By Key**. These are described below.

Every time you enter the Edit Window from the Database Editor, you must click on the *Next* gadget to display the first record of the database. If you want to browse sequentially, then choose *By Record #* using the Multistate gadget at the top right corner of the Edit Window. You can use the *Prev* and the *Next* gadgets to browse through and edit the data fields of each record, clicking on Update for each record to save your changes.

If you want to browse your records according to the key field order, then select *Browse by Key.* Do this by typing in one or more key values and then selecting the By Key option. For instance, in our example the key field was specified as the last name. If you want to find out the data in the records for a student named "Smith," type this name in the last name field and click on the *Next* gadget or the *Prev* gadget until you find the record. The data displayed in the fields will be the record containing the last name "Smith."

Delete is for deleting the currently selected record; i.e., the record whose fields are being displayed in the Edit Window.

If you have too many fields in your database to be displayed in the Edit Window, AmigaVision creates additional pages. If there is more than one page of fields in a single record, the current page number is indicated in the lower right. If there is more than one page, you may move between pages by using the *Prev Page* and *Next Page* gadgets at the bottom of the window.

When you are finished editing, click on the *Exit* gadget to exit the Edit Window and return to the Database Window. You can now create a new database or edit the keys and/or the data of an existing database. Or you can click on the *Exit* gadget to exit the Database Editor.

You may also create your own data entry forms as an alternative to the one provided in the Database Editor for data input by using the Data Form and Form Exit icons. For additional information, see the Database section in the AmigaVision Reference section.

Chapter 5

Questions and Answers

Chapter 5: Questions and Answers

This chapter tells you how to use AmigaVision to accomplish specific tasks. The information is presented in question-and-answer format and assumes that you have read Chapters 1 through 4 and understand how AmigaVision works.

1. How do I get started on a project?

As with any project, you need a clear objective for your application before using AmigaVision. You use the Flow Editor in AmigaVision to develop an outline of the sequence of actions. You then fill the outline with content items such as pictures, sound, music, text, and so on. Test the flow repeatedly and refine it until your objective is met.

It is a good practice to design your application from the "top down" by using the Module icons to specify the major modules in your application. Once you have a satisfactory design, then you can start implementing the modules from the "bottom up" by defining individual elements.

2. How do I synchronize videodisc motion and music (or digitized sound)?

The *Pause* gadget is intended for synchronization of dynamic events such as video motion, animation, digitized sound, music, and synthesized speech. When the *Pause* gadget is turned on, these events are executed and the application waits until each is completed. If Pause is OFF, then the events are started and the application executes the following event immediately.

Therefore, to synchronize video motion with digitized sound, use the **Video** icon to define the video motion and set the Pause to OFF. Then use the **Sound** icon to define the digitized sound. This approach gives a rough synchronization due to disk access time and the video device response time. If you want to synchronize sound precisely to a specific video frame, then you should use the following procedure:

Use the **Resource** icon to load the sound file prior to the Video icon. Then specify the **Video** icon as above and set the Pause to OFF. Next use the **Wait-Condition** icon. In this icon specify the condition through the Expression Editor using an expression such as:

Video() >= Frame-Number

where Frame-Number is the variable you must define to store the video frame number at which you want the sound to start and Video() is a standard function that returns the current video frame number.

Following the **Wait-Condition** icon, use the **Digitized Sound** icon to start playing the sound. You can adjust the value of the Frame-Number until you achieve precise synchronization.

Alternatively, you can start the sound first and then the video.

In the case of an animation you can use similar techniques for synchronization. There is a function called anim() in the Expression Editor that will return the animation frame number. You can also use **Timer** and **Delay** icons creatively to achieve synchronization in time rather than by specific frame number.

3. How do I draw lines and curves?

Use the Object Editor to specify static geometric shapes using the **Graphics** icon. For more complex drawings use a Paint package and import the file to AmigaVision.

4. How do I display the values of variables on the screen?

You can use the **Graphics** icon to create a Text object through the Object Editor. In the requester for the Text object you have a choice of specifying a variable name. During run time the value of this variable will be displayed at the location of the Text object. For instance, you can specify a Text Object "Name:" followed by the variable name NAME to display as follows:

Name: Smith

where Smith is the string value stored in the variable NAME.

You can also use Text objects in **Forms** to display values of variables.

5. How do I create Multilingual applications?

Typically, you need different text, font, speech and sound files for each language version. You can use variable names for filenames for those audio-visual elements that will change from one language version to another. Thus all the filenames associated with the Spanish version could have a prefix or suffix called SPA and the German version GER. When the application starts, you can ask the user to select the language of his or her choice from a menu. If the user chooses Spanish, then set a variable for the file prefix, say, PREF to be equal to the string SPA. Then you can use a filename with the PREF variable as shown below:

[PREF]-sound1.iff

where PREF is the variable prefix and **-sound1.iff** is a specific filename that follows the prefix.

You must have separate sound and other files for each of the language versions.

6. How do I create menus?

Menus have three components: the background image, the alternate options, and the actions that result from the choice. In AmigaVision you use a **Screen** icon to specify the background image followed by a **Wait-Mouse** icon to specify hit boxes. The **Wait-Mouse** icon is followed by a sequence of **If-Else** icons corresponding to each of the choices in the menu. The partner of each of the **If-Else** icons specifies the consequence for the choice. See the If-Else section for an example of a menu.

7. How do I create attract loops for information booths?

Attract loops are repeated animation or video motion segments for getting the attention of passers-by in a shopping mall or a store. Once a user is attracted, he or she should be able to exit the attract loop and browse through the information. After the user leaves the kiosk the attract loop is displayed again.

To create such a loop use a **Loop** icon and specify an Endless loop. As a child of the loop use the **Video** or the **Animation** icon that acts as the attract feature.

You must define an **Interrupt** icon prior to starting the attract loop. Inside the Interrupt you can define the menus and other modules containing the information for the consumer. If you use any Wait icons, each must be timed. In this way the presentation can return to the attract loop if a user simply walks away from the kiosk in the middle of a menu selection. You can accomplish this by using the Timeout feature of the Mouse Wait icon or by using a Group Wait icon.

8. How do I create input fields for typed text input? How do I create answer-judging routines for text input?

Input fields are standard objects used in specifying the **Forms** icon. You can use the **Forms** icon for any type of user input: string or number. The Input Fields limit the input to a maximum length of 255 characters. Thus, the longest text input you can accept is limited to 255.

To create answer-judging routines you can use the Input Fields to get user inputs and assign them into variables. You can use conditional tests using **If-Then** or **If-Else** icons by comparing the user input to correct answers pre-stored into a set of variables. You can also use wild card characters: ? for a single character and * for a string. These help you tolerate spelling mistakes or help look for a key word in a sentence.

9. Can I use different fonts for user input fields?

No. AmigaVision uses standard Amiga Intuition string gadgets for input fields. These have fixed size and background, and the user has no control over them.

10. How can I run other programs concurrently with AmigaVision during presentation?

Since the Amiga is a multitasking computer, you can run other programs simultaneously with AmigaVision. Those other programs can be started outside or inside of AmigaVision.

To start other programs outside of AmigaVision, you can run them using Workbench or from a SHELL.

To start programs from inside AmigaVision, use the **Execute** icon. The Execute icon can run programs in either Workbench or SHELL mode. Your AmigaVision flow is paused by the Execute icon until the external program finishes running unless you use the "Run" command in the SHELL mode. For example, if an Execute icon in SHELL mode performed the command "run yourprogram", AmigaVision would continue processing immediately after starting the other program. If the Execute icon simply performed the command "yourprogram", AmigaVision would pause until the other program finished.

Using the ARexx interface, it is possible for AmigaVision and another program to exchange information. Please refer to Appendix B for more information and examples.

11. How do I set up a glossary for the user?

Using the **Text File** icon to display ASCII text files through a window, you can create a glossary. First create a text file with an editor or word processor. Each definition in the glossary should start with a Unique String. This string is typically the word or the term which is being defined. Then you can search for this term in your text file and display the text following the term. The searching is done by specifying a string variable name in your Text File requester.

Typically in an application you will present the user with a list of the terms you have defined in the glossary. The user will select a term, and it gets stored in the string variable to be used for searching the glossary. The string could be something like: QUASARS. The text file will find the first occurrence of the string "QUASARS" and present the definition. Here we have used all upper case letters in QUASARS. The string search is case sensitive, so the search will ignore any previous mention of Quasars in the glossary. It is recommended that you use all upper case for the terms which are defined.

12. How do I develop Runtime modules for use on low memory systems?

The Amiga is a versatile computer that is ideal for multimedia applications. But multimedia components such as pictures and sound require a lot of memory for any reasonably useful application. Designing an application to run on all the machines requires very careful planning.

Among the things to do are:

- Use low resolution pictures.
- Use small animations & songs.
- Use few transitions.
- Don't preload files.
- Use synthesized speech over digitized sound when possible.
- Close Workbench.
- Test your application before distributing it.

13. How do I set AmigaVision to come up in interlace mode?

You have two options. The first is to invoke AmigaVision from CLI by typing **AV I** at the prompt. The second method is from the Workbench. Select the **AmigaVision** icon. Then select the **Info** option from the pull-down menu for Workbench. You will be presented with a requester where you add the term LACE to the Options list.

14. What is the maximum number of records, fields and key fields allowed in the database?

The number of records is limited only by disk size. You are allowed up to 128 fields, with a maximum of 4000 characters (total) per record. This means that although the maximum is 128 fields, you could use up all 4000 characters with 20 strings 200 characters in each length. You can have up to 100 characters indexed. Therefore, if you have fields which are 100 characters long, you cannot index other fields.

15. What specific steps should I take if I get the videodisc error message?

The first thing to check is that your videodisc player is on and that there is a videodisc loaded. Next check to make sure the cable is secure. In the Video Setup requester check the player and baud rate settings to ensure both the Amiga and videodisc player are talking at the same speed. Finally, if you still get the message, your cable may not be correct: some videodisc players use null modem cables while others use straight serial cables. Check your videodisc player owner's manual for cabling information.

16. Where does AmigaVision look for my instrument files when I try to play a music file?

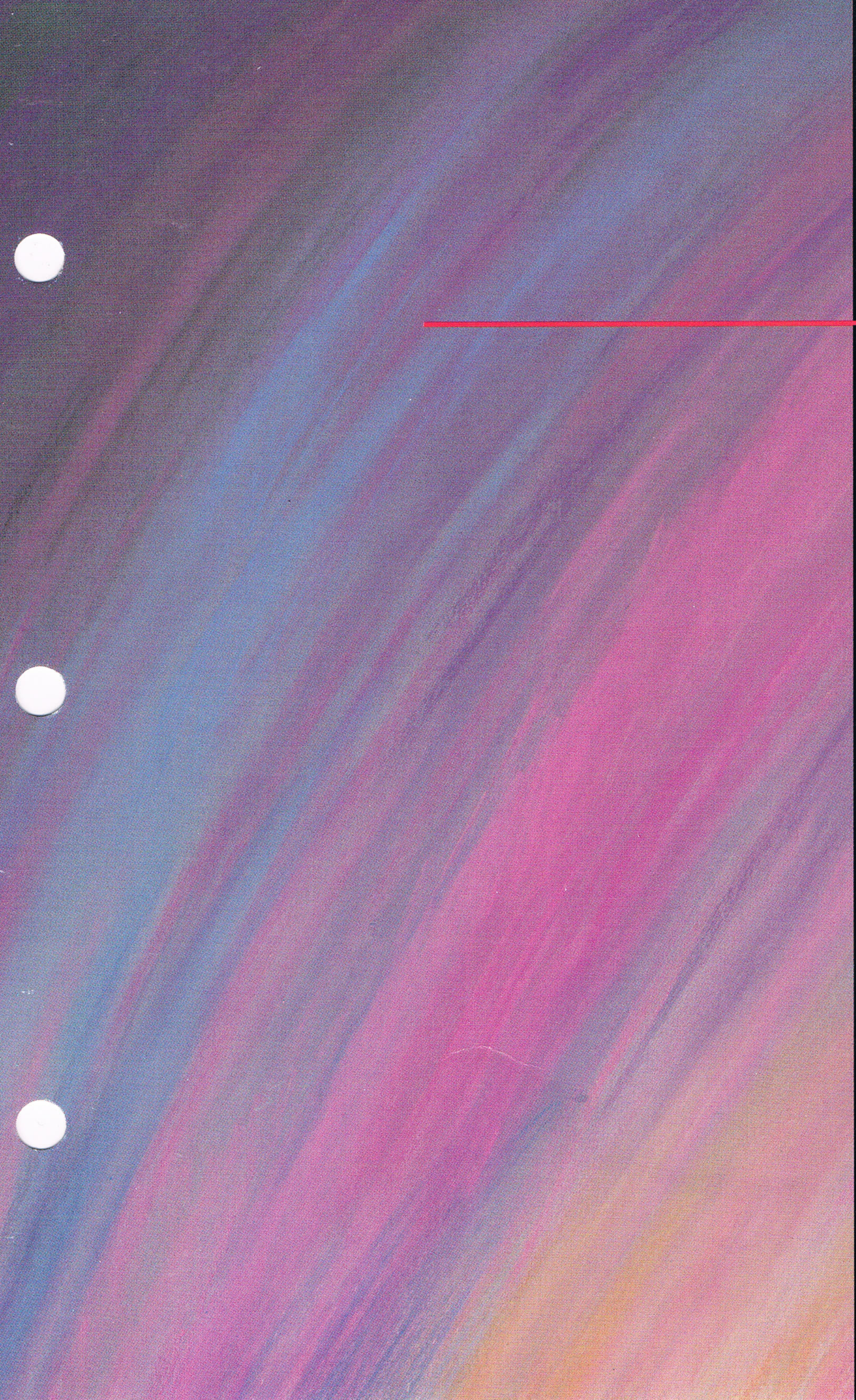
AmigaVision first looks in the path specified for Instruments in the Music requester. If this field is blank, AmigaVision looks in the directory specified in the Defaults requester. One of these two fields must be set to the path where the instrument files are stored.

17. How do I use local and global variables?

Modules and Subroutines contain variables which apply to that icon and its children. The **Variable** icon that is placed as a child of a **Module** or **Subroutine** creates local variables for that Module or Subroutine. A **Variable** icon that is placed before the first **Module** icon creates truly global variables that will appear in all variable lists throughout the flow.

18. How do I make sure all the necessary files are in my Runtime?

Instrument files should be loaded in the Resource icon if there is sufficient memory. Instrument files will only be copied automatically when they are loaded as Resources at some point in the flow. Font files must be individually copied to the Runtime disk. The fonts will later need to be copied into the font drawer of the user's system (usually in SYS:fonts).



User's Reference

USER'S REFERENCE

Chapter 6

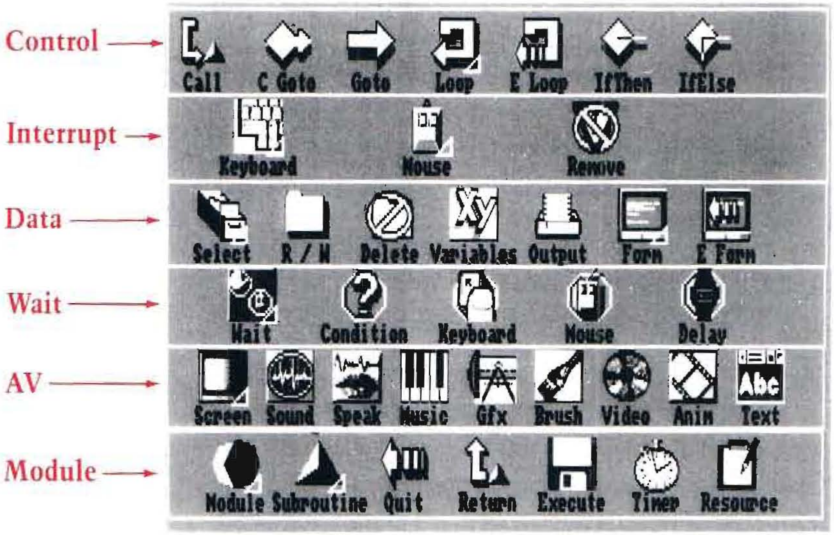
Icons and Requesters

Chapter 6: Icons and Requesters

This chapter deals with all six classes of icons and the requesters that accompany each class. Each class of icons is first described in general terms, then each is explained in detail.

First, here's a brief description of each class of submenu icons.

AmigaVision Submenu Icons



Control Icons — used to affect the flow of a project through use of branches and conditional statements.

User Interrupt Icons — used to place repetitive and global hit box events, such as keystrokes and mouse button clicking during a presentation.

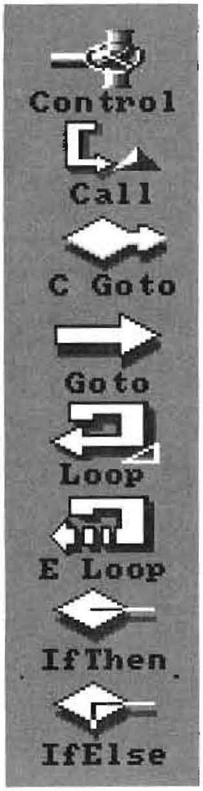
Database Icons — used to manipulate, add and remove information stored in a database which is used in a presentation.

Wait Icons — used to affect the flow of a project through use of pauses, such as waiting for a condition to become true before the flow continues.

Audio Visual Icons — used to place graphics, sound, animation, music, video and custom graphics objects into a presentation.

Module Icons — used to organize a flow into easily manageable units and control system functions.

Control Icons



Purpose

Control icons specify the logic that governs the flow of a project. Control icons include various kinds of *branching* and *looping* facilities.

Usage

There are three icons for branching on a condition, two for specifying loops, one for calling a subroutine, and one for unconditional branching.

These icons are useful for creating menus, multiple-choice tests, or similar flow structures.

The three types of icons for conditional branching are **If-Then** Icon, **If-Then-Else** Icon, and the **Conditional Goto** Icon.

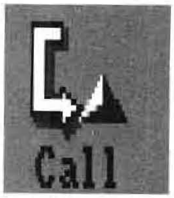
You must specify the conditions for branching in the form of an expression. The result of the condition (true or false) determines the flow of an application.

There is also a **Goto** icon which unconditionally branches from one icon to another.

Looping may also control the flow of a project. The **Loop** icon marks the beginning of a loop. The actions specified by the children of the Loop icon will be performed repeatedly until the loop is terminated. You can terminate the loop either by a count or a conditional expression. You also have the option of specifying endless loops. Use the **Loop Exit** icon to exit the loop explicitly from within a loop.

Another control icon is the **Call** icon. It changes the flow by referencing a *subroutine,* which is usually a stand-alone procedure located elsewhere in the outline.

Call Icon



Purpose

The **Call** icon executes a subroutine. A subroutine is an independent segment of the flow normally used repeatedly in an application.

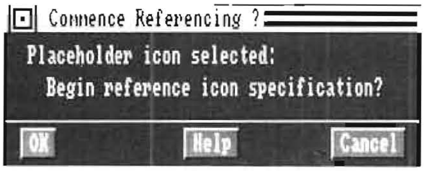
Relation

The **Call** icon cannot contain children, but requires a partner icon, which is a reference to a **Subroutine** icon in another part of the outline.

Usage

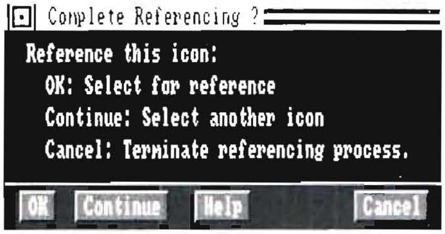
A subroutine is a collection of icons with a **Subroutine** icon as its parent. You may wish to create subroutines for sections of the application which will be used more than once. You use the **Call** icon in your flow to reference a subroutine to be executed.

When you place the **Call** icon in the outline, a placeholder for the partner automatically appears. Double-click on the **Placeholder** icon to begin the referencing process. This process allows you to scroll through the flow window and select the specific **Subroutine** icon to be referenced.



Commence Referencing ?

Placeholder icon selected:
Begin reference icon specification?

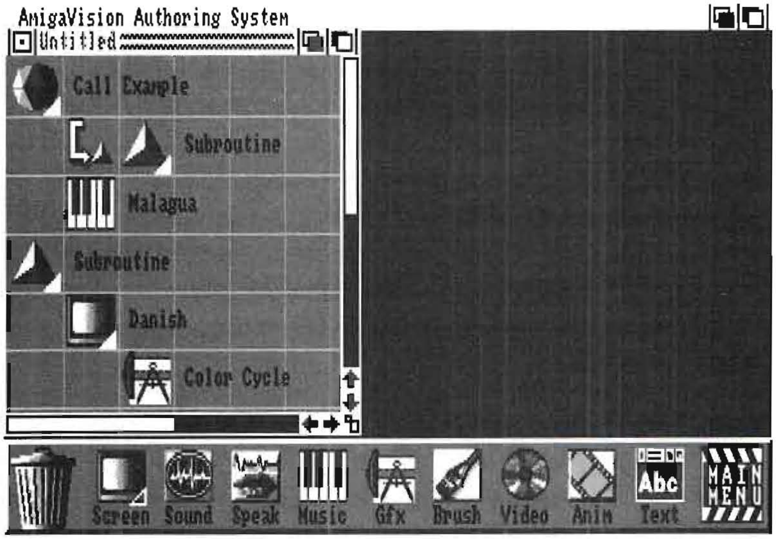


Complete Referencing ?

Reference this icon:
OK: Select for reference
Continue: Select another icon
Cancel: Terminate referencing process.

During a presentation, when an application reaches the Call icon, the referenced subroutine is performed. If either a Return icon is encountered or if the subroutine is completed, the application will continue in the outline starting with the icon following the Call icon.

Call Example



In this example, the Call icon branches to a subroutine which is placed elsewhere in the outline. The project calls a Subroutine 1 which displays a screen and color cycles it. After the subroutine is completed, the project returns to the icon following the Call icon and plays the music.

Conditional Goto Icon

Purpose

The **Conditional Goto** icon is used to branch to another part of the outline based on a specified condition. This icon conditionally transfers the flow of logic from one part of the application to another.

Relation

The Conditional Goto icon cannot contain children, but requires a partner. The partner is a reference to an icon elsewhere in the outline.

Usage

When you place the Conditional Goto icon in the outline, a placeholder for the partner automatically appears. Double-click on the placeholder and click *OK* to begin the referencing process. You can now scroll through the flow window and select the specific icon for referencing.

Note the following restrictions on the use of the Conditional Goto:

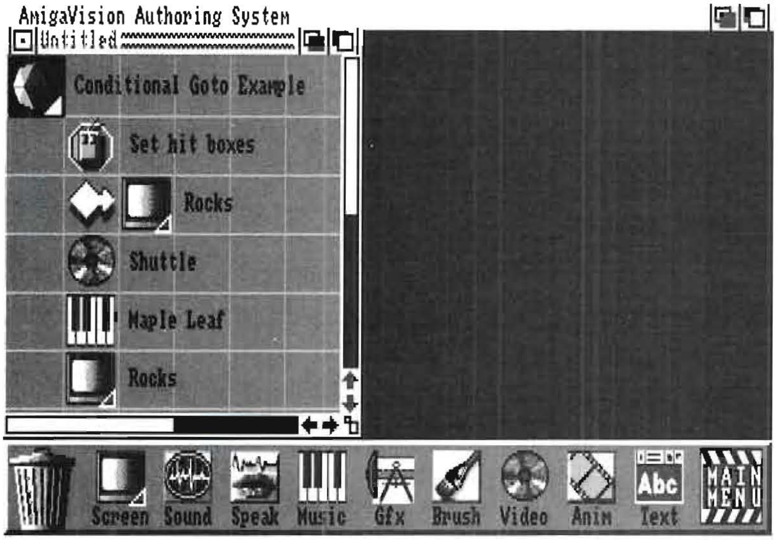
1. ***You may not use Conditional Goto icons to branch to any icons to the right of the Goto icon in the flow.***
2. ***The Conditional Goto icon cannot reference itself.***

To specify the condition for branching, double-click on the icon, which opens the Expression Editor.

During presentation, if the specified condition is true, the application branches to the referenced icon and continues from that point. If the condition is false, the icon following the Conditional Goto is executed.

The Conditional Goto icon offers a powerful way of creating branching in the Flow Window, but be careful while using it, as you may unintentionally create an infinite loop.

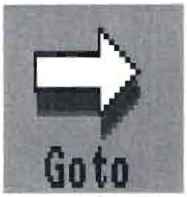
Conditional Goto Example


AmigaVision Authoring System
Untitled

- Conditional Goto Example
  - Set hit boxes
  - Rocks
  - Shuttle
  - Maple Leaf
  - Rocks

If the condition is true, the application branches to the referenced Screen icon. Note that after the Screen icon is executed, the application does not return to its original position in the outline, but continues with the icon following the Screen icon. If the condition is false, the application continues with the icon following the Conditional Goto, and displays a videodisc segment.

Goto Icon



Purpose

The Goto icon unconditionally branches or transfers control of flow to an icon in another part of the outline.

Relation

The Goto icon cannot contain children, but requires a partner. The partner is a reference to an icon in another part of the outline.

Usage

When you place the Goto icon in the outline, a placeholder for the partner automatically appears. Double-click on the placeholder which allows you to scroll through the flow window and select the specific icon to be referenced.

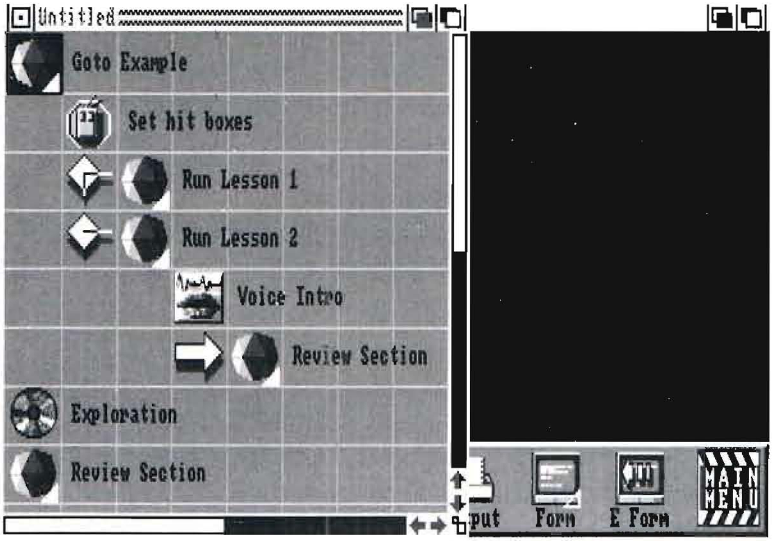
Note the following restrictions on the use of the Goto:

1. ***You cannot use a Goto icon to branch to any icons to the right of it.***
2. ***The Goto icon cannot reference itself.***

During presentation, when the application reaches the Goto icon, it unconditionally branches to the referenced icon and continues from that point.

Use care, as it is possible to create an unintentional infinite loop with this icon.

Goto Example



The Goto icon is best used to transfer control from one major section of a program to another (for instance, a new chapter in a course). In this example, the Goto icon is branching to a Module icon, representing an entirely different flow of icons.

Use the Goto icon only as a last resort, since liberal use of Goto icons is generally considered bad programming practice. Whenever possible, use Call and Subroutine icons to keep your programs manageable and make debugging easier.

Loop Icon

Purpose

The Loop icon is used to specify a loop structure. Its children define the actions to be performed within the loop.

Relation

The Loop can contain any other icons as children. It may also contain one or more loops called nested loops.

Usage

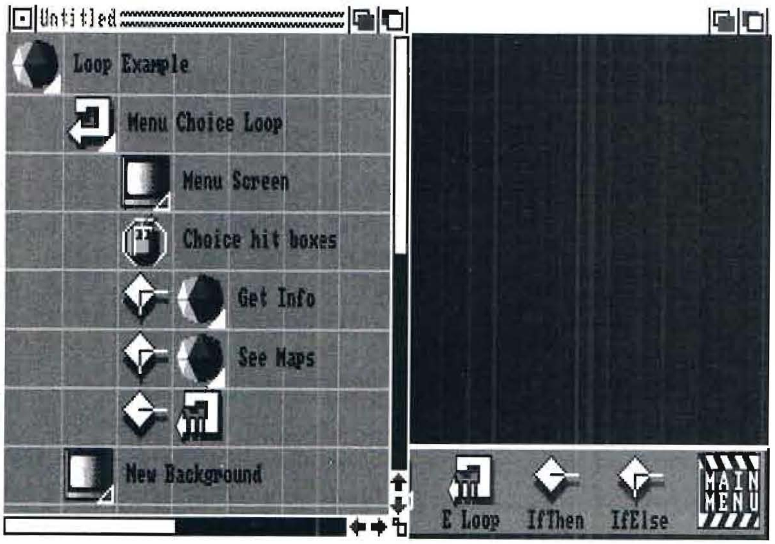
You use the Loop icon to set up the loop structure to cycle through a group of children icons.

Three types of loops may be set up: Endless, Counted and Conditional. The actions of children icons of the loop are performed repeatedly until an exit condition occurs. Each type of loop has a different exit condition.

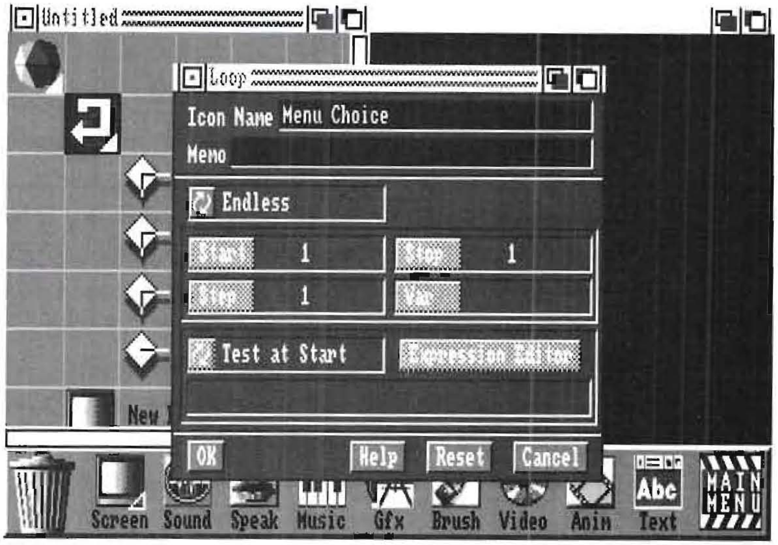
You can terminate an endless loop only by a Loop Exit icon. The Loop Exit icon can generally be used to terminate all types of loops. The counted loop terminates at the end of the count, and the conditional loop terminates if the specified conditional expression is evaluated as false. You have the additional option of specifying whether the condition for termination is evaluated at the beginning or at the end of each loop cycle.

During presentation, when an application reaches the Loop icon, the actions of the children icons are performed. When the actions of all the Loop's children are completed, the application will resume execution from the beginning of the loop. If an exit condition is reached, the loop stops and the application moves on to the next sibling of the Loop icon.

Loop Example



This example shows a typical use of the Loop icon: creating a menu. The first child of the Loop sets the graphics mode for the menu, and the Wait Mouse creates hit boxes with the menu choices. After the user clicks on any of the choices, the appropriate Module (each containing its own flow of AmigaVision icons) is performed. After the children of any one of those Modules have finished execution, control returns to the loop, and the process is repeated starting with the menu screen graphic.



Loop

Icon Name Menu Choice

Memo

Endless

Start 1

Stop 1

Step 1

Var

Test at Start

Loop Requester

Clicking on the Multistate gadget below the Memo field cycles through the available loop types: **Endless, Counted** and **Conditional**.

A **Counted** loop has three parameters: numeric values for **Start, Step** and **Stop.** These values can either be positive or negative. The Counter variable **Var** is optional. If you specify a variable name in this field, the **Var** will be assigned the current counter value at each iteration of the loop.

Start is the initial value of the counter and **Stop** is the final value. AmigaVision performs the actions of all the icons that are part of the Loop, then adjusts the value of the counter by the amount specified by **Step**. For instance, if **Start** is **2**, **Stop** is **10**, and **Step** is **2** then the loop will repeat for five times. If **Start** is **10**, **Stop** is **2** and **Step** is **-2** then the loop will repeat five times.

You may enter either numeric values or variables to specify **Start, Stop** and **Step** by clicking on their respective gadgets and entering the number in the **Specify Value Requester**.

The variable specified for Var is useful if you wish to evaluate expressions within the loop that change during each cycle of the loop. For example, you might want to set up a test in a loop that deducts points for each failed attempt. The **Var** could then be assigned to a variable called PENALTY. Within the loop you can use the Variables icon to specify an expression such as: **SCORE = SCORE - PENALTY**. When the loop starts, the PENALTY variable will have the same value as specified for **Start**. Let's assign a value of **10** for **Start**, **2** for **Step**, and **20** for **Stop**. During each cycle PENALTY will get incremented by 2, and there will be six repeats of the loop. If the user of the application repeats the test say, three times, the SCORE will then be whatever its initial value was minus 14.

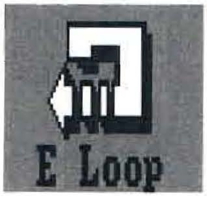
An *Endless* loop needs no other specifications. It simply runs continuously until interrupted by an interrupt event or a **Loop Exit** icon. If an interrupt event occurs, the loop will resume where it was interrupted after the interrupting event is completed. The **Loop Exit** icon lets you permanently exit a loop and proceed to the next sibling icon following the loop icon. You can also use either **Conditional Goto** or **Goto** icons to exit a loop and branch to a specific location in the flow.

When a Conditional loop has been selected, the **Test at Start/Test at End** gadget becomes active. Conditional (Test at Start) refers to a condition which is checked at the beginning of each loop cycle and which will end the loop if false.

Conditional (Test at End) is the same except the condition is checked at the end of each loop cycle. Therefore, this type of loop will always execute at least once. The loop continues to execute until the specified condition is no longer true. To specify the condition for either Start or End, click on the Expression Editor gadget to enter the Expression Editor.

The expression you create in the Expression Editor will appear in the text box below the Conditional (End) gadget after you exit the editor.

Loop Exit Icon



Purpose

The **Loop Exit** icon ends a loop. The presentation then continues with the next sibling icon following the loop.

Relation

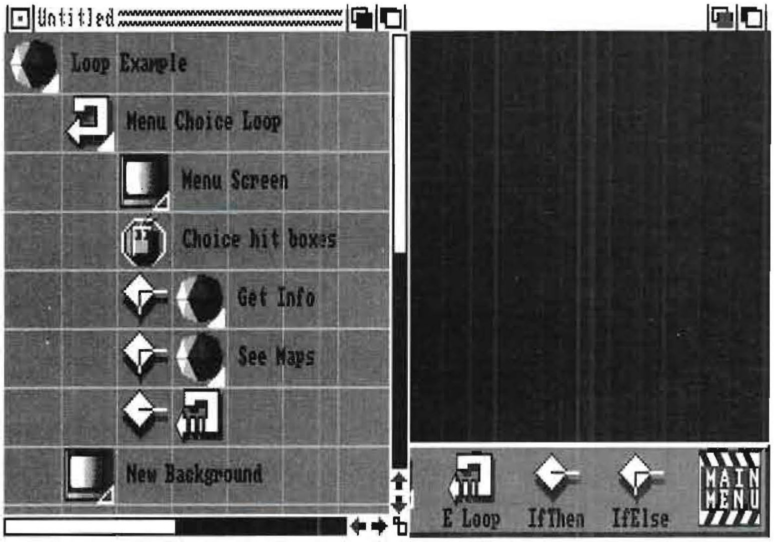
The **Loop Exit** icon cannot contain children.

Usage

When the presentation reaches the **Loop Exit** icon, the loop ends and the action of the Loop's next sibling icon is performed. The **Loop Exit** icon is ignored if it is placed outside of a loop.

Other ways to exit a loop are the **Conditional Goto** or **Goto** icons, which branch out of the loop.

Loop Exit Example



This is an example of an endless loop which will run repeatedly. The loop may, however, be exited conditionally. If box 3 is clicked, the loop will end. If this happens, the application continues with the Loop's next sibling. The **Loop Exit** icon can be used with any type of loop (endless, counted, conditional).

If-Then Icon

Purpose

The **If-Then** icon is used to define a condition which, if true, will cause the action of its partner icon to be performed.

Relation

The **If-Then** icon cannot contain children, but requires a partner.

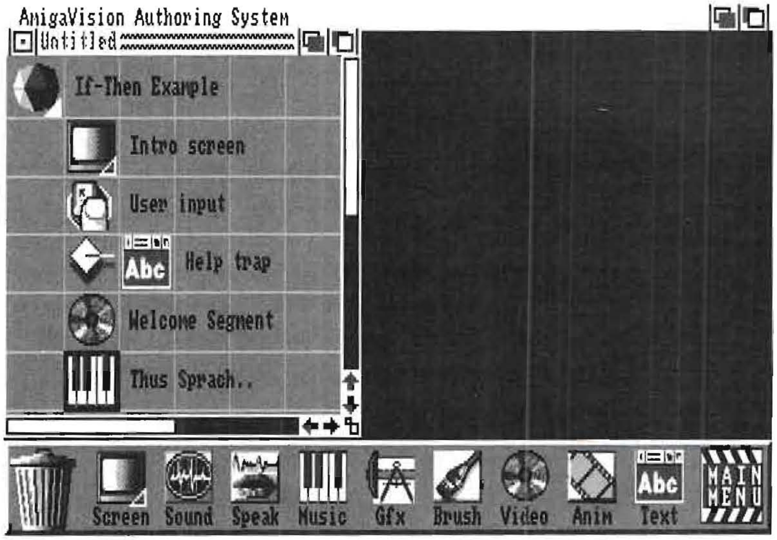
Usage

Double-click on the icon to specify the condition using the Expression Editor.

During presentation, if the condition specified for the icon is true, the actions of the partner icon will be performed and the application will then perform the action of the next sibling of the If-Then icon. If the condition is false, the partner icon will be skipped, and the action of the icon following the If-Then icon will be performed. In either case, the action of the icon following the If-Then is always performed.

Note that the partner of the If-Then cannot have siblings. To obtain the effect of siblings, use a parent icon, such as a Module icon, as a partner to the If-Then icon and add child icons to the Module icon.

If-Then Example



If the response from the Wait Keyboard is "Help," the action of the partner is performed (a text file is displayed). If the response is not "Help," the action of the next icon in the sequence is performed (playing a videodisc segment). The action of the icon just below the If-Then is always performed regardless of the response.

If-Then-Else Icon

Purpose

The **If-Then-Else** icon defines a condition for executing one of two separate icons: one for the case when the specified condition is true and one when it is false.

Relation

The **If-Then-Else** icon cannot contain children, but requires a partner.

Usage

Double-click on the icon to specify the condition in the Expression Editor.

During presentation, if the condition specified for the If-Then-Else icon is *true,* the application performs the action of its partner icon. It skips the icon following the If-Then-Else, which represents the Else part.

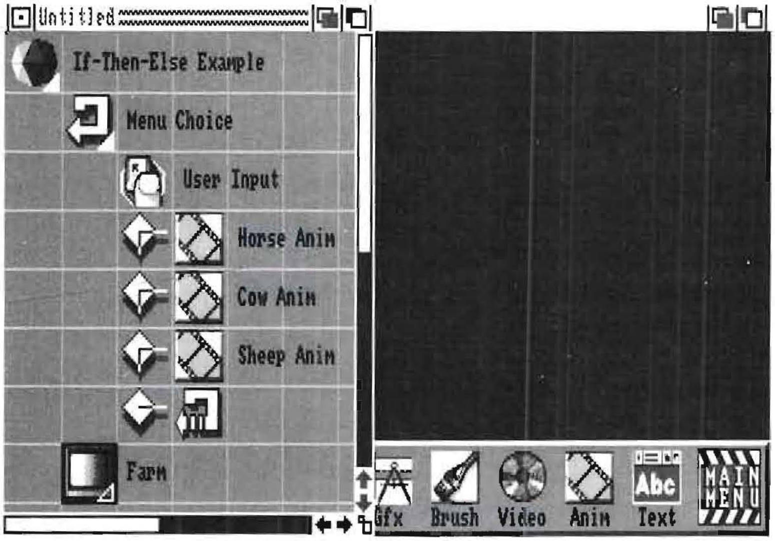
If the condition is *false,* then the application performs the action of the Else part, which is the icon that is immediately after the If-Then-Else icon.

You can use several If-Then-Else icons in a sequence to test for more than one condition. In effect, you can specify the Else part of an If-Then-Else icon to be another If-Then-Else icon. In such a case when a true condition is found, all the If-Then-Else icons that follow are skipped along with the very last else icon. This feature makes the If-Then-Else icon very useful in creating menus and multiple choice tests. It allows the application to check which response out of several specified ones occurred and perform the specified actions accordingly.

Note that the partner of an If-Then-Else icon cannot have siblings. To obtain the effect of siblings, use a parent icon such as a Module as a partner and then add children to the Module icon.

The difference between the If-Then icon and the If-Then-Else icon is that the If-Then icon always performs the actions of the icon that follows it, while the If-Then-Else icon performs either the action of the partner icon or the icon following it.

If-Then-Else Example

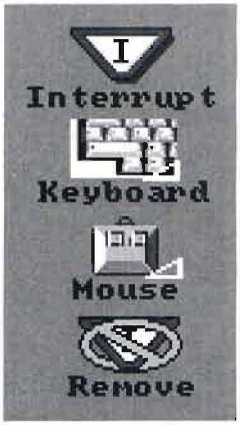


This example demonstrates the construction of a menu using the Loop icon, three If-Then-Else icons, and an If-Then icon. If the first condition is true, Horse Anim will be played. If it is not true, the next condition will be checked. Each condition will be evaluated similarly.

On the other hand, if the first condition is true, the other three icons will be skipped, and the application will begin the loop again and display the menu choices.

The condition specified in the Expression Editor uses the response() function to check the current response string set by the Wait Keyboard icon.

User Interrupt Icons



Purpose

The User Interrupt icons define actions when an application is interrupted during its presentation. There are three types of interrupt icons:

- **Mouse Interrupt.** When a mouse button click interrupts the presentation.
- **Keyboard Interrupt.** When a keystroke interrupts the presentation.
- **Remove Interrupt.** Clears or disables one or more interrupts.

Usage

There are several situations in which you might want to suspend the flow of an application. For instance, a teacher who wants to allow a student access to help might define a mouse hit box on the screen which says "Help." Any time the student clicks on "Help," the presentation will pause, and helpful information will be presented. The teacher can define the help information using a set of icons as children of the interrupt icon.

Common user interrupt inputs are the Esc key, Help key, and the function keys on the keyboard.

In this section an interrupt is called "active" when the application allows access to it. It is called "inactive" after it has been removed or terminated. For instance, an Esc key becomes an active interrupt as long as its action is defined and it is not terminated by the termination of its parent event or by explicit removal in the flow.

An interrupt is referred to as executing when it is invoked in an application and the actions of its children are being performed.

All Interrupts are active only at and below the level at which they are defined. This means that if an interrupt is placed as a child to another icon, it will only be active while the actions of other children of that parent are being performed. If you want an interrupt to be active through an entire application, you must define it as the first child to the topmost Module that is the parent of the entire application. Interrupts are disabled during text window and grouped wait execution.

Note the following when using interrupts:

1. ***If an interrupt is placed as a child to a module icon, it will be active while the actions of the other children of that module are being performed. As soon as the Module icon is completely presented, the interrupt will be automatically disabled. All visible attributes of the interrupt, such as hit boxes or display objects, will be removed.***
2. ***If more than one interrupt uses the same key, only the most recently enabled interrupt will be active.***

During presentation, if an interrupt is invoked by a keypress or mouse click, the application will pause and the actions of the interrupt's children will be performed. While the actions are executing, the interrupt is disabled to avoid recursive calling of the interrupt. For example, if the user presses Help for more information, he will be presented with help messages using icons as the children of the interrupt. During the presentation of this help information, the user will not be able to press Help again until he is finished viewing the help screen.

After the actions of the interrupt's children are completed, the application returns to the icon which was interrupted.

You can define the screen appearance, text and hit boxes for an interrupt using the Object Editor. Each Interrupt icon has a specific set of objects, such as hit boxes or text, that is exclusively associated with it. If the interrupt is active then these objects will appear on the screen.

There are two classes of objects when using an interrupt: Specific and Other. *Specific* objects refer to those added by the interrupt that is currently active. *Other* objects refer to all other objects that are displayed on the current screen that are not associated with the interrupt.

You may specify whether you want AmigaVision to not remove, temporarily remove, or permanently remove Specific objects from the screen during the execution of the interrupt.

The Multistate gadget choices are for Specific objects in the Keyboard and Mouse Interrupt requesters.

For example, during the execution of an application you may have a hit box with the words "Click Here For Help." As soon as the interrupt is activated, this box can be made to disappear by choosing the option of temporarily removing the specific object.

You also have the option to not remove, permanently remove, or to temporarily remove Other objects from the screen (those previously specified with the WAIT icons, Graphics icons, etc.). Note that objects which have been stamped into the background cannot be removed (see Graphics requester description for background objects).

When icons with time duration such as sound, music or animation are interrupted, the action is paused. After the actions specific to the interrupt are performed, then the paused action will resume from the beginning. Thus a music file or an animation will start all over again. You cannot resume an animation or musical segment at the exact place where interrupt occurred.

One exception to this rule is the playing of a videodisc segment because videodisc frames can be referenced directly. You can store the starting and the ending frame numbers for the segment in variables. If an interrupt occurred, the first action of the interrupt would be to pause the video motion. After the execution of interrupt actions, the application might provide the user with the options "Resume" or "Repeat." If the user wants the segment repeated, then variables can be used to repeat it. Or if the user wants to resume at the frame he had interrupted the application, then the application could simply play the rest of the segment. This is a commonly used technique in kiosk applications where a video segment is played over endlessly to attract passers-by. When someone interrupts the segment by touching the screen or by other methods of input, the video pauses and presents a menu.

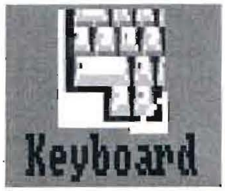
You have to be very careful when using interrupts. If you alter the background of the screen inside an interrupt (i.e., by displaying visuals with the interrupt's children icons) then you must restore the background screen when the interrupt actions are completed. You must remember that this background restoration must be done in a very general way, as the interrupt can be activated at many different parts of the application flow.

For example, imagine that you have a series of picture files to be displayed on the screen. Now suppose that you have an interrupt represented by a hit box on the bottom left hand corner of the screen with the message "Click Here For Help." You have defined the interrupt to put up a new background picture called HELP.PIC on the screen. When an interrupt occurs while one of your pictures is being displayed, then HELP.PIC will obliterate this background. To restore the screen background to the picture that was being displayed at the time of the interrupt, use variables to store the filenames of the pictures.

You might, for instance, use a variable called FILENAME to store the name of any picture that you are about to display. Then the last action item in the interrupt will be to restore the background to its former state by using the picture file stored in the variable.

If you do not want to go through this procedure, then avoid changing the background using picture, animation or brush files inside of an interrupt. You can simply use objects that are defined using the Object Editor.

Keyboard Interrupt Icon



Purpose

You can use the Keyboard Interrupt icon to allow an interruption to the application flow when certain keys are pressed. If one of the specified keys is pressed, the application will pause, and the actions of the children of the keyboard interrupt icon will be performed.

Relation

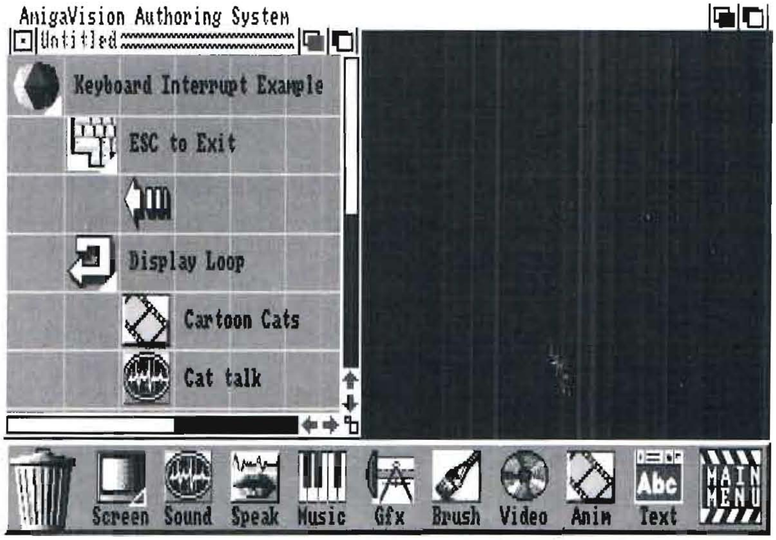
The Keyboard Interrupt icon can contain any icon as a child.

Usage

See the preceding sections for general descriptions and constraints on the use of interrupts.

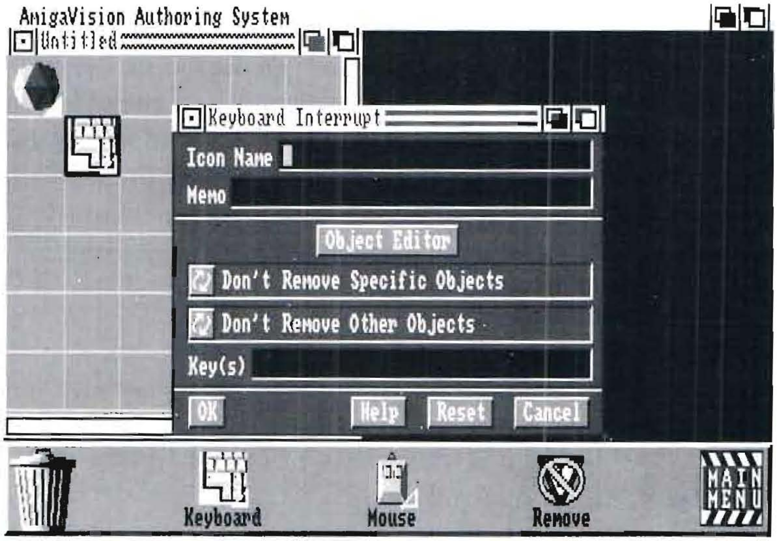
You use the Keyboard Interrupt icon to define specific keys, such as function keys which, when pressed during the presentation of an application, will interrupt the flow of the presentation. You may define a keyboard interrupt which is active through the entire application or only for a part of it.

Keyboard Interrupt Example



In this example, the text "Press Esc to exit" is placed on the screen. If the Escape (Esc) key is pressed while the animation or sound is played, the interrupt will occur, and the presentation will be exited. If the Escape key is not pressed, the animation and sound will be displayed repeatedly as defined inside the loop.

Keyboard Interrupt Requester



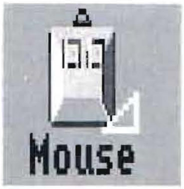
Click on the **Keys** field and type in the key(s) which will be used to cause the interrupt. If special keys (e.g., function keys, Help key, or the Escape key) are to be used, type the name of the keys as they appear on the keyboard in this field. The available special keys are **F1** through **F10, Esc, Del** and **Help**. For certain other keys such as the arrow keys you can enter **Up, Down, Right, Left, Enter, Return** and **Tab** for specifying these as interrupt keys.

Key combinations (such as Ctrl-A or Alt-F or the Right-Amiga keys) are not supported.

If you wish to have multiple interrupt keys defined for the same interrupt, then separate the keys by commas in the key definition field. For instance, you can type in **F1, Help, Esc** to specify that all these three keys enable the same interrupt.

Note: The keyboard interrupt icon only responds to *single keystrokes;* therefore you cannot, for example, specify a word to be typed in that will act as an interrupt.

Click on the **Object Editor** gadget to open the Object Editor where you may create text or objects to be placed on the screen with the interrupt. For instance, you might want to inform the user "Press Help key for additional help or Esc key to exit." You can do this by using text objects. These objects are uniquely associated with the interrupt, and whenever the interrupt is active they will appear on the screen.



Mouse Interrupt Icon

Purpose

You can use the Mouse Interrupt icon to define an interrupt to the application flow if the mouse is clicked in a certain area of the screen called a hit box. If interrupted, the application will pause and the actions of children of the interrupt icon will be performed.

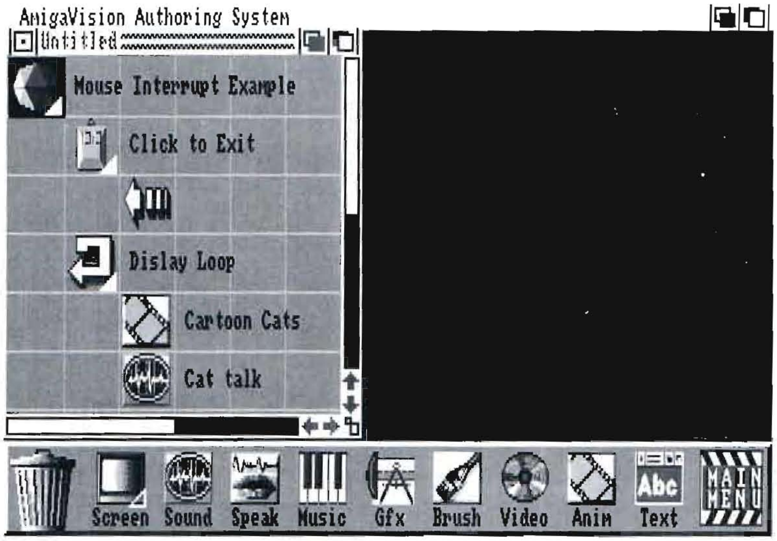
Relation

The Mouse Interrupt icon can contain any icon as a child.

Usage

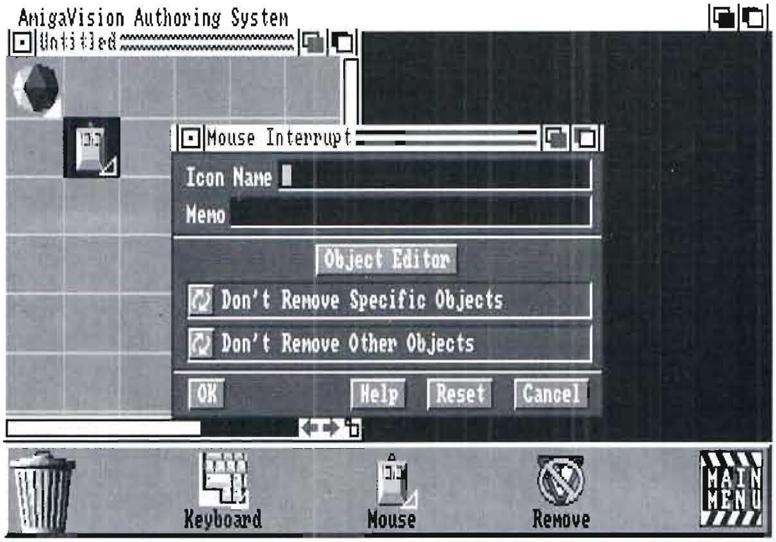
You use the Mouse Interrupt icon to define areas on the screen called hit boxes which, when clicked on with the mouse, will cause an interrupt in the program flow. You may define a mouse interrupt which is active through the entire application or only a part of it.

Mouse Interrupt Example



In this example, the text "Click to Exit" is placed on the screen. If the mouse is clicked at anytime during the presentation, the interrupt will occur, and the presentation will exit.

Mouse Interrupt Requester



Click on the Object Editor gadget to open the Object Editor where you may create text or objects to be used as hit boxes with the interrupt. These objects are uniquely associated with the interrupt, and they will be placed on the screen if the interrupt is active.

You create the hit box in the Object Editor using the Add-Polygon or Text/Var menu option. In the Object Editor you specify the string "Exit" as a response for the hit box. When the user clicks inside of the rectangle in your application, then the response string will be stored in the response() function described in the Expression Editor section of Chapter 4. The creation of mouse hit boxes in the Object Editor is also explained in Chapter 4.

Remove Icon

Purpose

The Remove icon is used to disable previously defined interrupts.

Relation

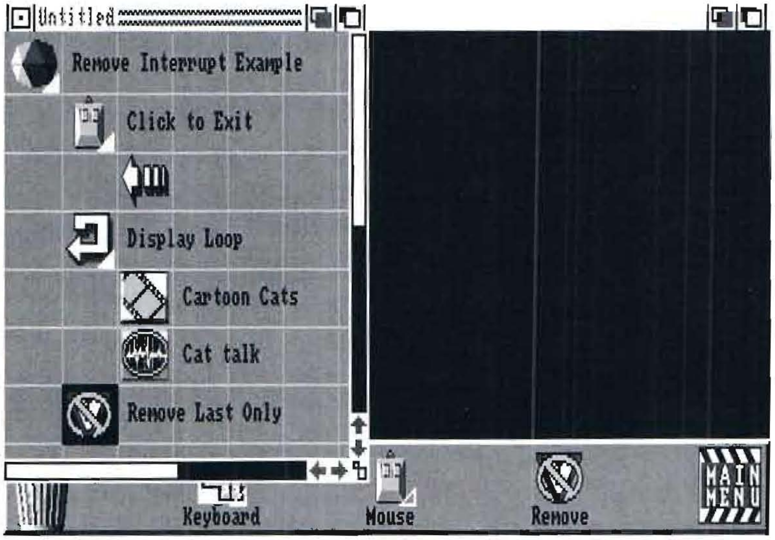
The Remove icon cannot contain children.

Usage

The Remove icon only removes interrupts at its own level. You may use the Remove to disable either the most recent interrupt or all of the interrupts which are siblings to the Remove icon.

If used as a child of an interrupt, the Remove icon will have no effect.

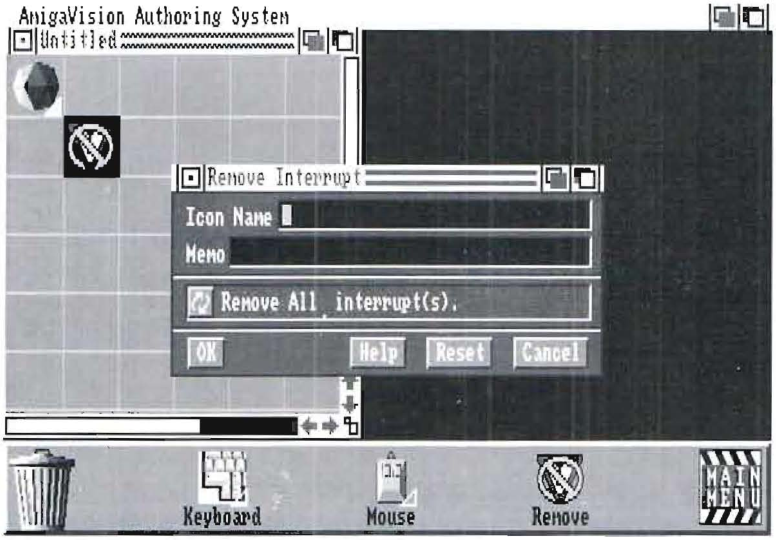
Remove Example



Remove Interrupt Example
Click to Exit
Display Loop
Cartoon Cats
Cat talk
Remove Last Only

In this example, the interrupt is placed in the main level of the outline. This means that it will remain throughout the application unless removed or re-defined. The Remove icon takes away all the interrupts.

Remove Requester



AmigaVision Authoring System

Untitled

Remove Interrupt

Icon Name

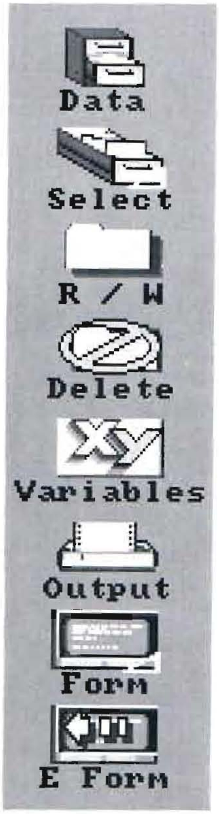
Memo

Remove All interrupt(s).

Click on the **All/Last** gadget to select the appropriate action. The **All** option removes all interrupts that have been defined prior to the Remove icon and which appear on the same level (the same vertical column in the flow) as the Remove icon. The Last option removes the most recent interrupt defined prior to the Remove icon and which is on the same level as the Remove icon.

Database Icons

Purpose



You can use the Database icons to define variables, define data-entry forms, store and retrieve data from a database, and define printed or file output in your application.

The database file doesn't have to be one created in AmigaVision; it can be any dBASEIII-compatible file. It is assumed here that you are familiar with database terms such as records, fields, keys, data files, and index files as defined in Chapter 4.

Usage

There are three icons that exclusively relate to database operations on an existing database:

- **Select**
- **Read/Write**
- **Delete**

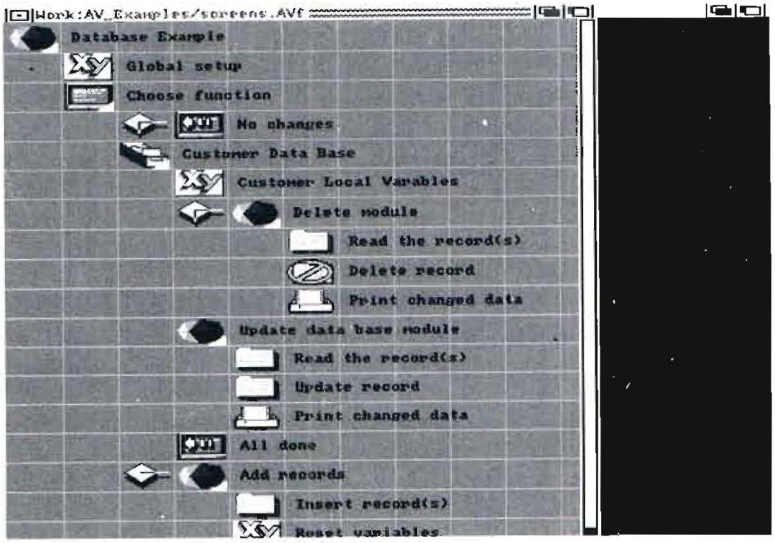
You can use these icons to select specific records in a database for reading into variables and for updating them. You can also insert or delete records. Use of these three icons assumes that a database file has already been created.

Prior to using the database icons in your application flow, please follow these steps:

1. ***Be sure you have a database created. If not follow instructions in Chapter 4.***
2. ***Using either the Module, Subroutine or Variable icons, define variables in which you will collect data for saving in your database.***

The example shown below uses all the database icons to illustrate their close relationship. Each of the icons in this section will refer to a segment of this example. In this example, the following task is accomplished:

A data-entry person has to go through the current customer database and add new customers, delete the ones who are no longer customers, and update information on existing customers. When he is done with each record he gets a printed output listing the information on that customer. When he is done with all the customers, he exits the application.



```
Work:AV_Examples/screens.AVf
Database Example
    Global setup
    Choose function
        No changes
        Customer Data Base
            Customer Local Varables
                Delete module
                    Read the record(s)
                    Delete record
                    Print changed data
            Update data base module
                Read the record(s)
                Update record
                Print changed data
            All done
        Add records
            Insert record(s)
            Reset variables
```

Select Record Icon

Purpose

The Select Record icon is used to open a dBaseIII-compatible database file and select records using one or more key fields. Records and key fields are explained in Chapter 4.

You must use this icon if you plan to read or modify your database files. You can simultaneously select records from a maximum of ten (10) separate database files. The Select record icon only locates records within the database; it does not read the record itself.

Relation

The Select Record icon can contain other icons as children.

Usage

You may only select records from dBaseIII-compatible database files using fields for which you have specified keys using AmigaVision's Database Editor.

A key is made up of one or more fields from the database and is used when searching the data file for a specific record or a set of records. You may want to organize a database of employee information alphabetically by last name or by employee ID number. When you create the database, you designate those fields (LASTNAME and EMPID) as keys. This lets you access your data records either by specifying an employee ID number or by their last name. If you use only the last name to search the database, then there is likely to be more than one record matching the last name, particularly if the name is a common one.

Before selecting records using one or more of the key fields, you must define a set of variables containing the values you wish to specify as the key fields. If, for example, you want to select all the records with the last name "Smith," you must create a string variable: VARNAME = "Smith". You can define variables using either the Module, Subroutine or Variable icons.

Then, using the Select Record icon, first enter the database filename and press the Enter key on your keyboard. This will open the database and display all the fields highlighting the key fields with a > character. You can double-click on the key fields, and a list of variables will be presented for you to link variables to key fields for searching the database.

For instance, the variable VARNAME in the above example can be assigned to the key field LASTNAME. Since you have specified VARNAME to be equal to the string "Smith," all the records where the value of the variable matches the contents of the field will be selected. Thus records containing either Smith as last names, or Smithsons, or any record with a last name whose first five characters match Smith will be selected. If you want to restrict the search to just Smith, then you must terminate your variable value with the | character; i.e., VARNAME = "Smith|". This will ignore all records with last names that do not exactly match Smith.

Also it is important to note that the key values such as 'Smith' are case sensitive. If your database contains last names with the first letter capitalized, i.e., 'Smith' rather than 'smith,' then your key value for searching must also be 'Smith'.

The Select Record icon can be a parent to the other database icons which perform the read and write operations. As a parent the Select Record icon acts like a loop; its children are performed repeatedly until all the selected records have been processed. In our example above, a Select Record icon will process all records with the last name that matches the key value of Smith.

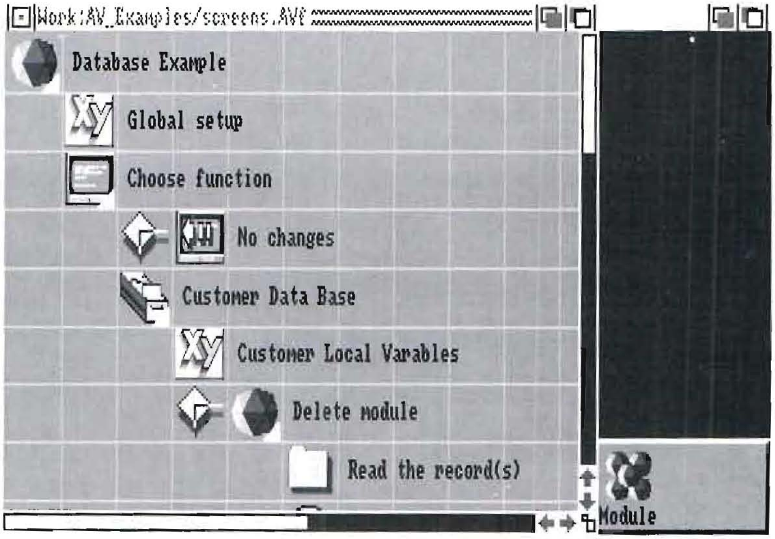
If you do not link any variables to key fields, the Select Record icon selects the very first record in your database. As a parent it then loops through all the data records sequentially. If the Select Record icon is not a parent, then it selects only the first record for use by its siblings.

Note that you may have as many as ten separate database files open for selection in your application. However, since each Select Record icon can access only one database, each of the database files requires a separate Select Record icon. This is useful in a case where you have stored the names and addresses in one file and the test scores or other vital statistics in another file. By using two separate Select Record icons, you can select and read information from both the files.

If you use Select Record icon to open an eleventh database file, it will be ignored.

Once a database file is opened by the Select Record icon, the file is automatically closed by AmigaVision at the end of executing an application.

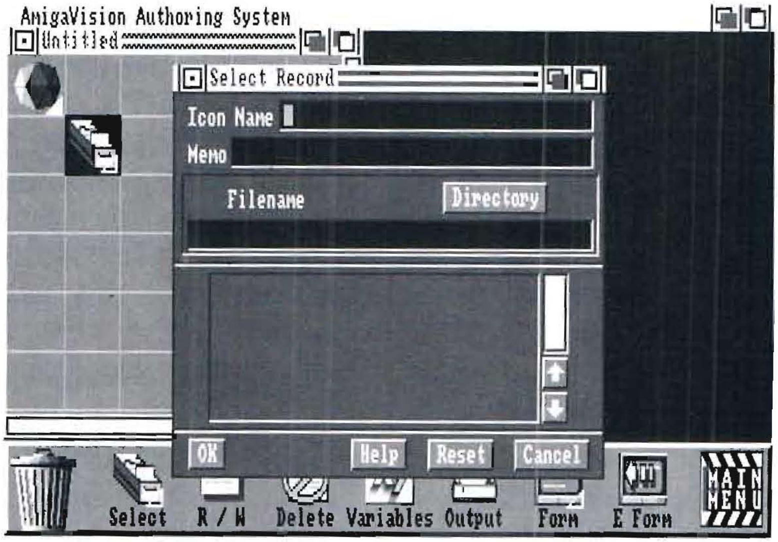
Select Record Example



The Module and Variables icons are used to define variables that will be used in record selection and will be read from the customer database. The Form icon is used to specify the last name, first name, customer number, and zip code, stores these values in variables. In the Form icon there are four hit boxes: Delete, Update, Add and Exit. If the user selects Exit then the application is exited, otherwise the control is passed on to the children of the Form icon. The first child is the Select Record icon. Normally, after the user hits Enter in each field, the cursor jumps to the next entry, and then exits after the last field is filled out. Be aware that in order to give the user a chance to exit or cancel from entering data, you must add your own hit boxes.

The Select Record icon uses the last name and the customer number as the two key fields for locating a record. The Select Record Icon opens and searches the database name called Customer and uses the key fields to find the record that matches both the last name and the customer number. If such a record is found, the actions of the Select Record icon's children are performed. If not, the action of the Select Record icon's sibling is performed.

Select Record Requester



Click on the **Directory** gadget to select a database through the file requester, or type the database name of a previously created database in the filename field and press the Enter key.

The specified file's fields are shown in the window with the key fields marked by the > character. Clicking on the field name in the list presents a list of variables that have been defined in the flow. Select a variable from the list to link it to a field. Repeat this procedure for each of the fields you want to link to a variable. Be sure that the variables you are using have been assigned values prior to the Select Record icon. When the Select Record icon executes, it first locates one or more records using the specified key fields and then uses the non-key fields, if any, to locate the record precisely.

If a key field is not assigned a variable, the interpretation is SELECT ALL the records that match that key. If no fields are assigned to variables, all records are selected starting from the first one.

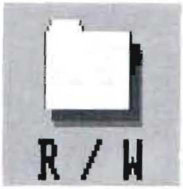
If more than one key field is linked to variables, the interpretation is SELECT ALL records that match key field 1 and key field 2 and key field 3 and so on.

For example, if you link the key field LASTNAME to the variable name VARNAME and STATE to VARSTATE, and you have defined VARNAME="Smith|" and VARSTATE= "VA", then you will select all the records for employees named Smith who live in VA. You may also use non-key fields in addition to the key fields to select records; however using only key fields is faster. If you use non-key fields when specifying a search, it takes much longer to perform the search.

If the Select Record icon fails to find a single record in the database that matches the specified keys, then it will not perform the actions of its children and the application will move to the next sibling icon. If you want to perform specific actions, such as adding a new record, then you must use variables to determine if the Select Record icon failed to locate a record.

In our above example, just prior to the Select Record icon, define a variable (using the Variables icon) called VARNAME = "NONE". Then use the Select Record icon to select the record. Following the Select Record icon use a Read/Write icon to link the variable VARNAME to the LASTNAME field and read the record. The reading operation will fail if no record has been selected. You can use a conditional icon such as If-Then or If-Then-Else to test if VARNAME = = "NONE". If it is TRUE then you can assume that the Select Record icon failed to locate a record.

Read/Write Record Icon



Purpose

The Read/Write icon reads from and writes to database records which were previously selected using the Select Record icon. The database file must be compatible with the dBaseIII file standard. It could either have been created using the AmigaVision Database Editor or other commercially available database software.

Relation

The Read/Write icon cannot contain children.

Usage

You use the Read/Write icon to read data from the fields in the currently selected database record into variables and to write data from variables into record fields. When you use the icon, you assign fields to variables, and select the appropriate action (either **Read, Insert** or **Update** records).

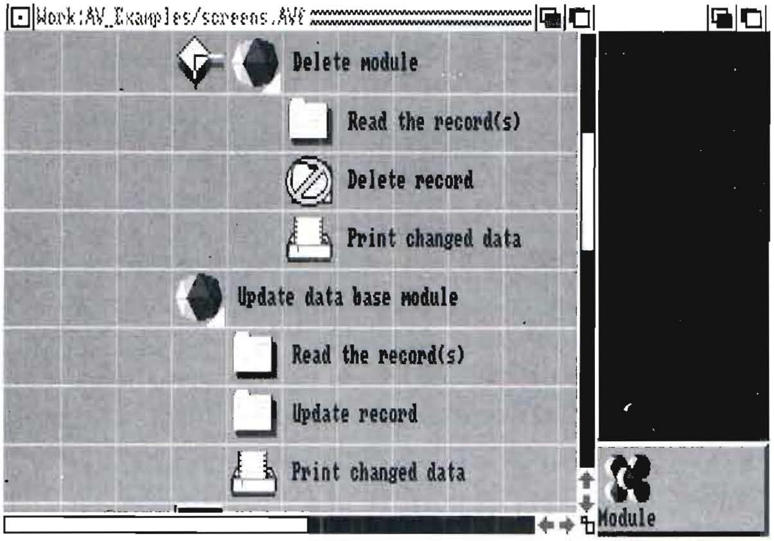
In an AmigaVision presentation this icon is the primary method for updating information in a single record in the database and can be used in the following way. Imagine that you have created a database of employee information which contains the field SALARY. You want to update the SALARY field to reflect a 5 percent pay increase.

1. *Use the Variables icon to set up a variable called MEMSAL and one called NEWSAL.*
2. *Use the Select Record icon to select the first record.*
3. *Use the Read/Write icon to read the contents of the SALARY field into the variable MEMSAL.*
4. *Now you will need another Variables icon to calculate NEWSAL: NEWSAL = MEMSAL * 1.05.*

5. ***To complete the process, use another Read/Write icon to update the value of the variable NEWSAL to the field SALARY in the current record of the database.***

   Note that this will be an update operation, since you are replacing existing values in your data file.

Read/Write Icon Example



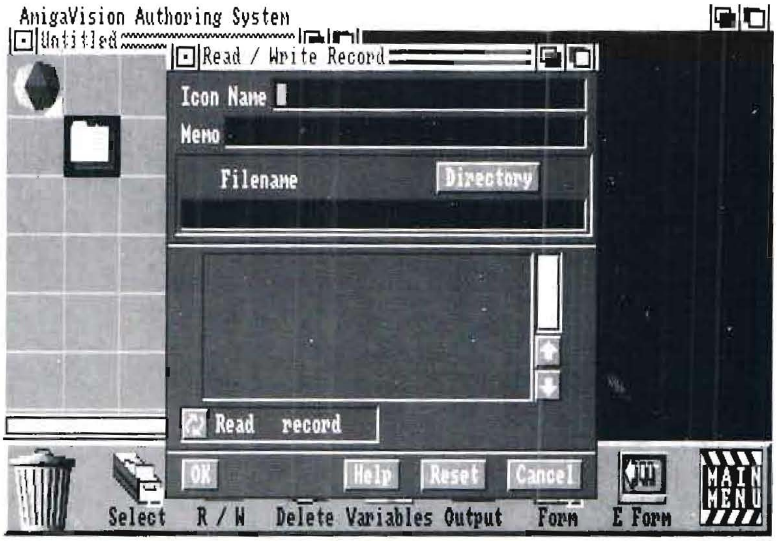
If the Select Record icon succeeds in finding a record that matches the last name and the customer number, then the application branches to its children. The first child is a Variables icon that sets the Boolean variable called "select" to the TRUE state. Then the application branches to the Read/Write icon, where the fields in selected records is read into the variables that are linked.

If the data-entry person selected the option of deleting in the Form icon, then the application branches to a Read/Write icon, which reads the record fields into variables for printing later. It then deletes the record.

Likewise, Read/Write icons are used in the flow for updating the fields in the record with values entered in the form.

Finally, if there is no record found by the Select Record icon and the data-entry person selects the Insert option in the Form icon, then a Read/Write icon is used for inserting a new record.

Read/Write Requester



Specify the name of the database file to be used in the filename field. You may type the filename or click on the *Directory* gadget to open the File requester. Be sure that the database file you have specified has been opened previously by a Select Record icon. If you do not have a Select Record icon defined, then your Read/Write icon will fail to do anything.

Click on the Multistate gadget to select one of three operations: **Read, Insert** or **Update** a record. **Read** is used for reading the data fields in the currently selected record into the variables that have been linked.

If the operation is an **Update**, the icon will update the values in the specified fields of the currently selected record.

The **Insert** option adds new records to the database. You can use it, for instance, if the Select Record operation fails to find a record with key fields matching key values specified.

A list of the available fields is shown in the window. Double-clicking on the field name in the list presents a list of previously defined variables. You can select variables from this list to link them to the fields. The linking operation is identical to that in the Select Record icon. It is a good practice to use different variables for record selection and for record reading/writing. For instance, you may want to use multiple Read/Write icons for reading the same record into different variables in your application.

Delete Icon

Purpose

The Delete icon removes a currently selected record which was previously selected using the SelectRecord icon.

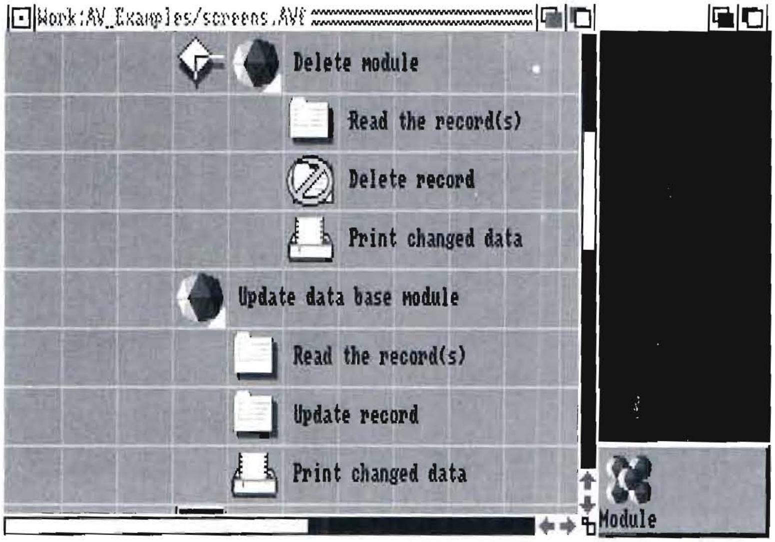
Relation

The Delete icon cannot contain children.

Usage

You use the Delete icon to remove a record from the database. Records to be deleted must first be selected with the Select Record icon.

Delete Example



Work:AV_Examples/screens.AVE

Delete module

Read the record(s)

Delete record

Print changed data

Update data base module

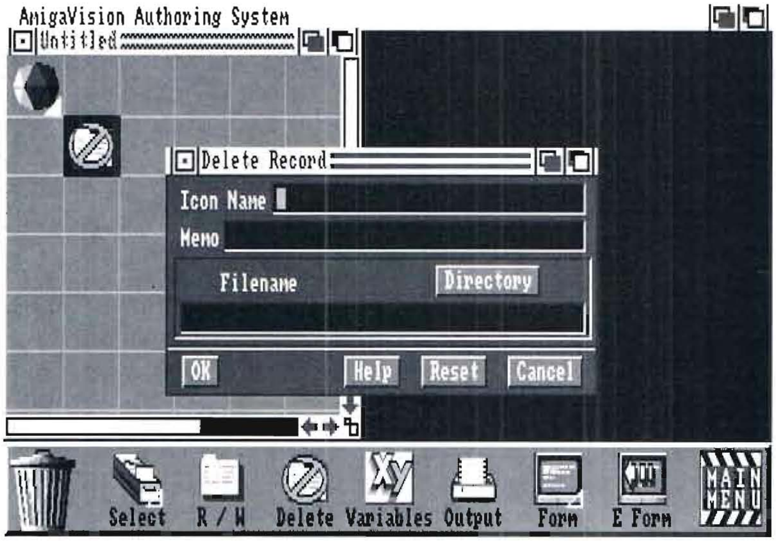
Read the record(s)

Update record

Print changed data

In the example, because the Delete icon is a child of the Select icon, all records that match the selection key values are deleted. If you want to delete just the first record that matches the selection criteria, then as in the case of the Read/Write icon, you must place the Delete icon as the sibling of the Select icon.

Delete Requester



Specify the name of the database file to be used in the *Filename* field. You may type the filename or click on the *Directory* gadget to open the File requester.

Variables Icon

Purpose

You use the Variables icon to define new *global* variables, assign new values to variables, or create expressions. Global variables can be accessed throughout an application once defined. See Chapter 5 for more information on global variables.

Relation

The Variables icon cannot contain children.

Usage

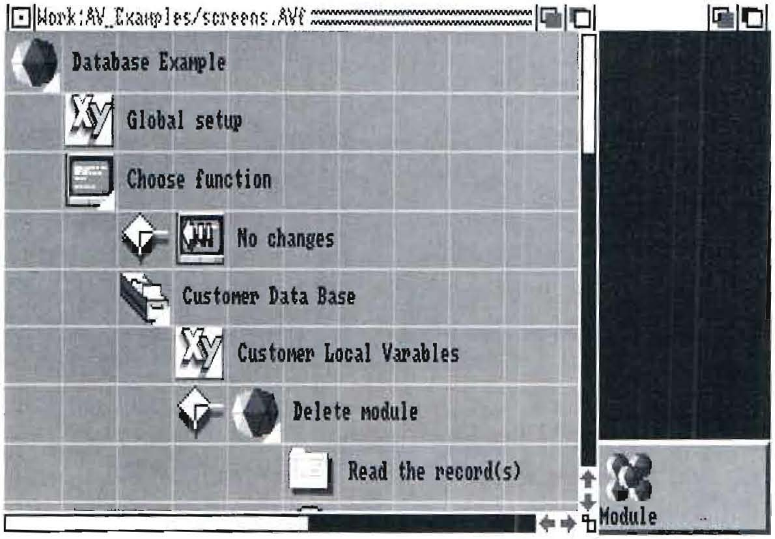
You may use the Variables icon to define variables by assigning values or expressions to them.

In AmigaVision, variables can be created using a Module icon, a Subroutine icon, or the Variables icon. Variables created by Module and Subroutine icons are called *local.* For instance, a variable defined in a Module icon will be useful only within that Module. Similarly a variable defined in a Subroutine icon has validity only within that Subroutine.

The Variables icon, in contrast, can create variables that can be used throughout an application. Thus they are called *global* variables. In addition to creating global variables, the Variables icon can be used for modifying the values of even the local variables created with the Module and Subroutine icons.

If you use variables that have not been previously defined, new global variables will be created.

Variables Icon Example



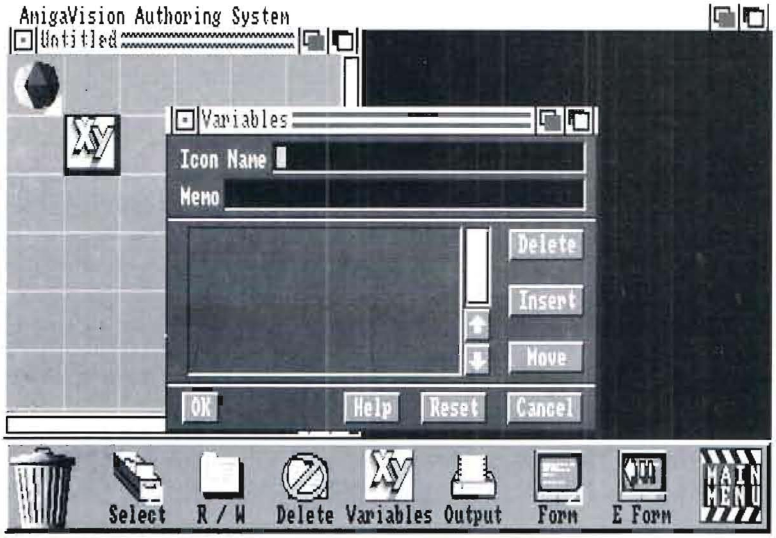
In the example from the previous section the Variables icon is used initially to define a set of *global* variables such as:

lastname = ""
AccountNumber = 0

etc. (P!ease refer to the section on the Expression Editor in Chapter 4 for more information about global and local variables.)

These variables are used in the example for reading and writing to database records. The fields in the records are linked to the variables using the Select Record and Read/Write icons. Also, the Form icon links the input data fields to variables. Thus, variables offer a powerful method of collecting and storing information.

Variables Requester



The **Delete, Insert** and **Move** gadgets are used to arrange expressions in the Expressions (or variable definitions) Window. When you click on Insert, the Expression Editor is presented and you may define a number of variables. When you exit the Expression Editor by clicking on the *OK* gadget, the variables along with their defining expressions will appear in the Expressions Window.

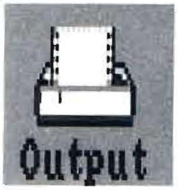
To insert expressions in a particular place in the list of existing expressions in the window, click on the expression below which you wish to insert. When the expression is highlighted, click on the *Insert* gadget.

To delete an expression from the list, click on the expression in the window. Then click on the *Delete* gadget.

To move an expression elsewhere in the list, click on the expression. Click on the *Move* gadget. To move a variable definition above its original position, click on the definition above the one you want to move. To move a variable definition below its original position, click on the definition below the one you want to move.

Double-clicking on a variable definition presents it in the Expression Editor for you to edit.

Output Icon



Purpose

The Output icon is used to send a single line of output to a disk file or a printer.

Relation

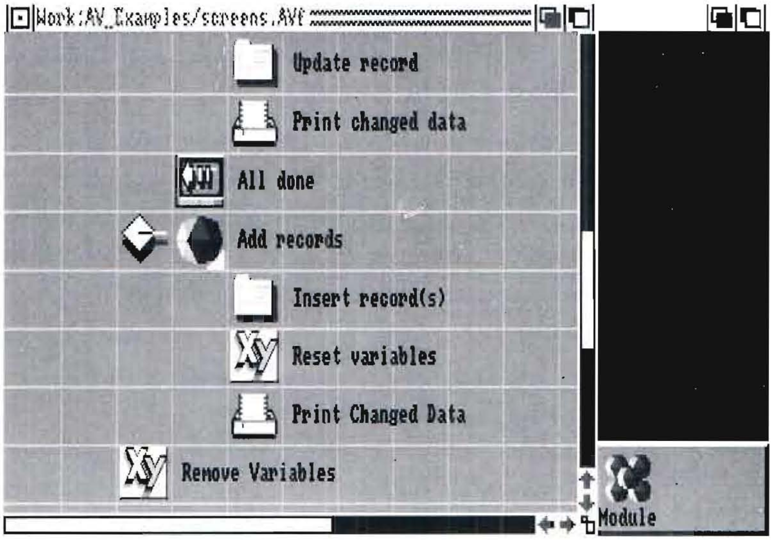
The Output icon cannot contain children.

Usage

You may use the Output icon to produce a line of formatted output for values of variables in your application. The output can be sent either to a file or to a printer. If the output is sent to a file, it is saved in the ASCII format.

The Output icon allows you to format only a single line of output with a maximum of 132 characters. If you need multiple lines (for example, to include headings in a report), you must use one Output icon for each line.

Output Icon Example

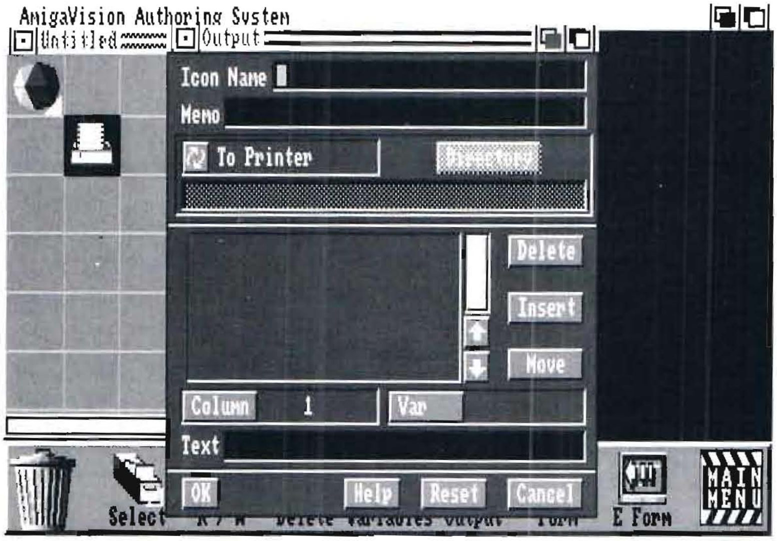


In the example, the selected records are formatted for output to a file which is later displayed with the Text File icon.

Output Requester

Specify the name of the output file to be used in the *Filename* field. You may type the filename or click on the *Directory* gadget to open the File requester.

Click on the *Multistate* gadget to specify the output destination: *Printer* or *File.*



To Printer

Column 1

If you select the File option then an ASCII file is created using the specified filename. If a file by that name already exists, the output is simply appended to the end of the file.

When you click on the *Var* gadget the variable list is presented. When you select a variable from the list, it will appear in the *Var* field. When the Output icon executes, the value of this variable will be printed at the column location that you specify.

You have the option of specifying a text string instead of a variable. Click on the *Text* field to type in a literal string of text to be output. This is useful for specifying a title, column heading, or other fixed textual information on the line to be printed.

After specifying a variable or a text string to be output, you have to click on the *Column* gadget to enter the starting column number for printing. The maximum number of columns is 132. You should carefully plan the output so that string variables such as names have adequate space. The Specify Value requester is presented. There you can enter the column number. Thus you have to allocate all the variable values you want shown in a single line over the limited space of 132 columns.

When you have specified the variable or text string and the corresponding column location, click on the *Insert* gadget to add the specification to the output format window. This window shows the starting column location; a code letter (T or V) indicating whether at that location a text string or a variable value is being printed; and either the name of the variable or the text string.

The *Delete, Insert* and *Move* gadgets are used to arrange the output format items in the window to the left of them.

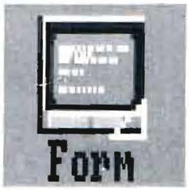
To insert a new item at a particular place in the list in the window, click on the item below which you wish to insert. When the item is highlighted, click on the *Insert* gadget.

To delete an item from the list, click on the item and then click on the *Delete* gadget.

To move an item elsewhere in the list, click on the item and then click on the *Move* gadget. Then to place the item above its original position, click on the item above which you want to move. Or to place the item below its original position, click on the item below which you want to move.

Double-clicking on an item presents it for modification in the *Var* (or *Text)* and the *Column* fields. After modification you must insert it again using *Insert* gadget. Use the *Delete* gadget to delete the original item.

Data Form Icon



Purpose

The Data Form icon defines forms on the screen for data entry by users during the execution of the presentation.

Relation

The Data Form icon can contain other icons as children.

Usage

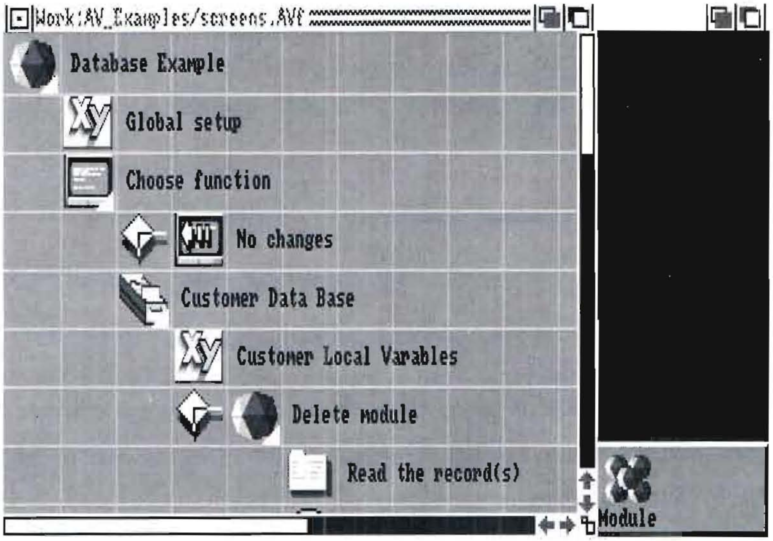
You use the Data Form icon to create forms on the screen for data entry. A data entry form is a screen which contains data fields into which the user types information. The data fields for the form are created in the Object Editor.

In the data field object requester, define the field type and length. Available field types are shown in the Database Editor section of Chapter 4.

In addition to data fields, the form may include hit boxes for entering non-field responses. As an example, the form might include hit boxes for OK and CANCEL, which could be used to exit the form.

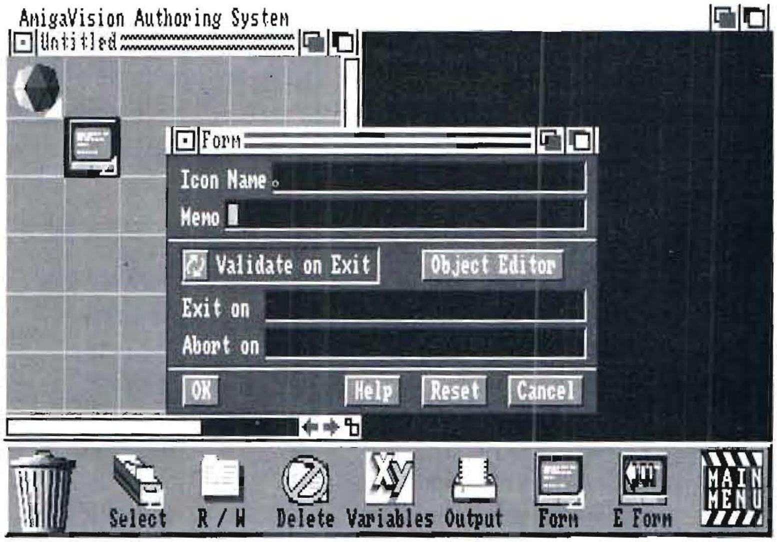
You can validate the data entered by the user in a form by checking each field. The Object Editor lets you specify validation expressions and error messages for each field. Alternatively, you may validate the data entered in all the fields after the user exits the form. You may also specify response strings for exiting and/or canceling a form.

Data Form Icon Example



This is an example data form created in the Object Editor. The form has three fields: NAME, SSN, and COURSE. It also contains a hit box with the response string "OK."

Data Form Requester



Click on the *Object Editor* gadget to enter the Object Editor. In the Object Editor, you define all the data fields for the form. For each of the data fields you must specify a variable in the Object Editor. The data entered by the user and the variables in your application are thus linked. The value of the linked variable gets updated as the data is entered for each field.

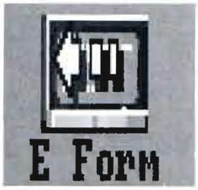
In the Object Editor you can also specify hit boxes for exiting or aborting the form.

Click on the Multistate gadget to specify whether input should be validated for each field or when the form is exited. If you choose *Validate Fields,* the exit condition for each field is checked as the data for the field is entered. If you select *Validate on Exit,* field entries are not validated until the form is exited.

Note that if you select validate on Exit, you must specify a string in the *Exit On* field for the form. This string must match the response string you specified for hit box when you created the data form in the Object Editor. Similarly, if you want to provide the user with an abort option in the data form, you must specify a hit box with a response string. This same string must then be entered in the Forms requester in the field for the *Abort On* string.

A form is exited normally when the user exits the form by clicking on the hit box with the response string specified in *Exit On*. The values of variables that are linked to the data fields are then modified and saved. However, if the Abort condition occurs, variables are reset to their values prior to the Data Form icon.

Click in the *Exit On* field and type in the string which will cause the form to be exited. Click in the *Abort On* field and type in the string which will cause the form to be aborted.



Form Exit Icon

Purpose

The Form Exit icon is used to exit or abort a data form operation.

Relation

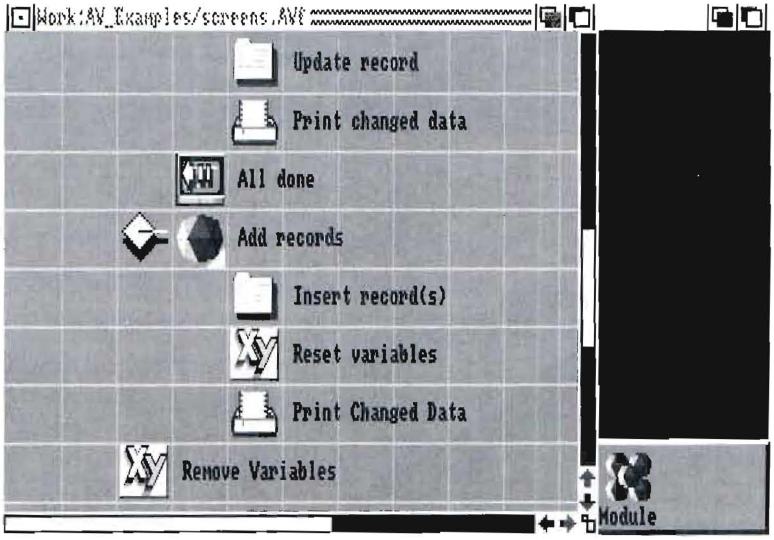
The Form Exit icon cannot contain children. The Form Exit icon can only be used as a child to the Data Form icon.

Usage

You use the Form Exit icon to exit from a data form without the use of a hit box. The Form Exit icon is not required for each Data Form icon. Normally, a data form is exited when either the *Exit On* string or the *Abort On* string is encountered. You may, however, include hit boxes for other response strings on the form. You might then check for one these other response strings and conditionally execute other icons. The Form Exit icon could then be used to exit the Form icon.

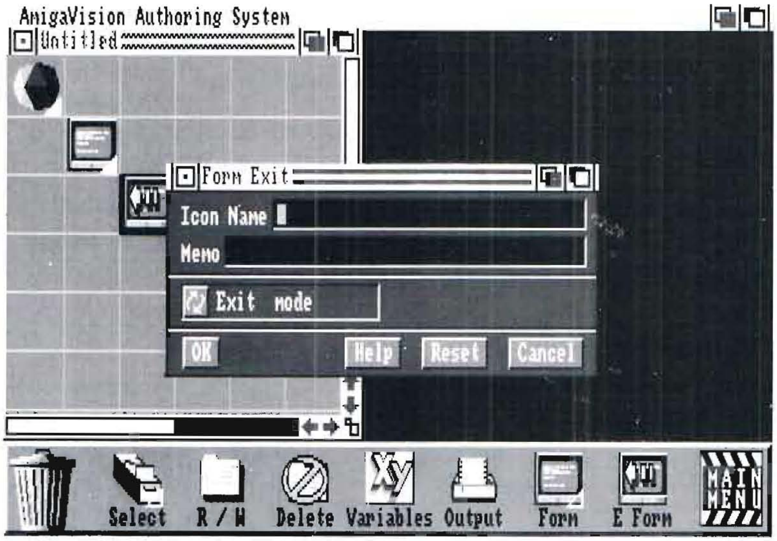
Using the Multistate gadget, you may specify the mode as either Exit or Abort. If you specify Exit, the variable values adjusted by the data fields in the form are modified and saved. If you choose Abort, variable values are reset to their values prior to the Data Form icon.

Form Exit Example



In this example, the Form Exit icon allows an exit from the data form from a child icon.

Form Exit Requester



Form Exit

Icon Name

Memo

Exit mode

You may specify either *Exit mode* or *Abort mode* with the Multi-state gadget. When *Exit mode* is selected, variables adjusted by the form are modified and saved. If you choose Abort, variables are reset to their values prior to the Form icon.

Wait Icons

Purpose

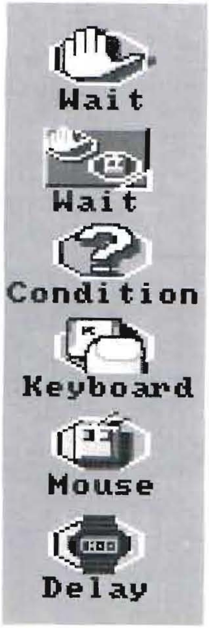
The Wait icons cause the program to halt execution until a desired user response or a specified condition occurs.

Usage

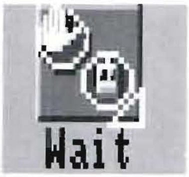
You use the Wait icons to pause the program and wait for input from the user. For example, you may wait for a mouse click or a key to be pressed.

You also use a Wait icon to wait for a condition to occur. For example, you may wish to wait for a timer to reach a certain point before proceeding.

You may combine Wait icons with the Grouped Wait icon which uses the logical operators AND and OR.



Grouped Wait Icon



Purpose

The Grouped Wait icon is used to combine Wait icons.

Relation

The Grouped Wait icon must contain children. It can only contain other Wait icons as children. It may not contain another Grouped Wait Icon as a child.

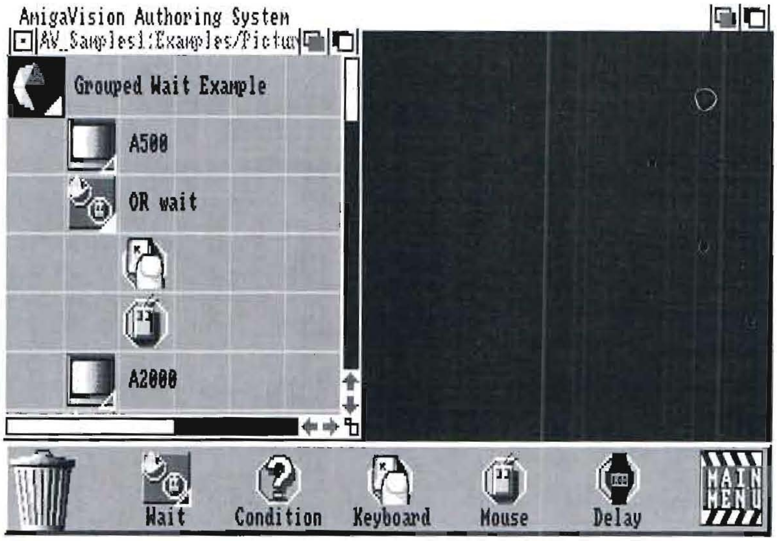
Usage

The only function of the Grouped Wait icon is as a parent to other specific Wait icons. The Grouped Wait itself does not cause an application to pause. It combines Wait icons using the logical operators AND and OR. For example, it can be used when you wish to wait for either a mouse response *or* a keyboard response. If one of these responses is received, the application will continue. This is useful when you want to allow the user to respond either through the keyboard or through mouse input.

In certain cases you might want to have input from both keyboard *and* mouse required in your application. You can specify this by choosing the AND option. The application will wait for both responses, but they may occur in any order.

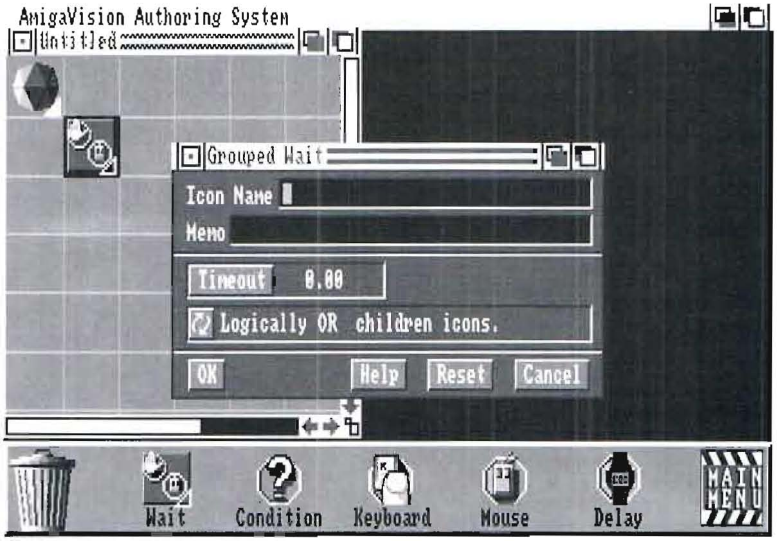
Without grouping under the Grouped Wait icon, the Wait icons will cause the program to stop and wait for each response to be given in order.

Grouped Wait Icon Example



In this example, the program waits for either a keyboard or a mouse response. When either response is given, the application proceeds with the Wait's next sibling, the Screen Icon.

Grouped Wait Requester



Icon Name

Memo

Timeout 0.00

Logically OR children icons.

Click on the Timeout field to specify the amount of time in seconds the presentation will wait in the event the desired response does not occur. This will open the Specify Value requester. If the Timeout is greater than 0, the program waits the specified number of seconds.

Note that the Timeout specified for the Grouped Wait overrides any and all timeout settings on the individual children Wait icons.

Click on the Multistate gadget below Timeout to choose either *AND* or *OR*. If Logically AND children icons are specified, the presentation will wait until all of the children's responses occur (in no specific order). If Logically OR children icons are selected, the presentation waits for either one of the children's responses to occur.

Wait Condition Icon

Purpose

The Wait Condition icon is used to wait for a specific condition to be true. Once the condition occurs, the application continues.

Relation

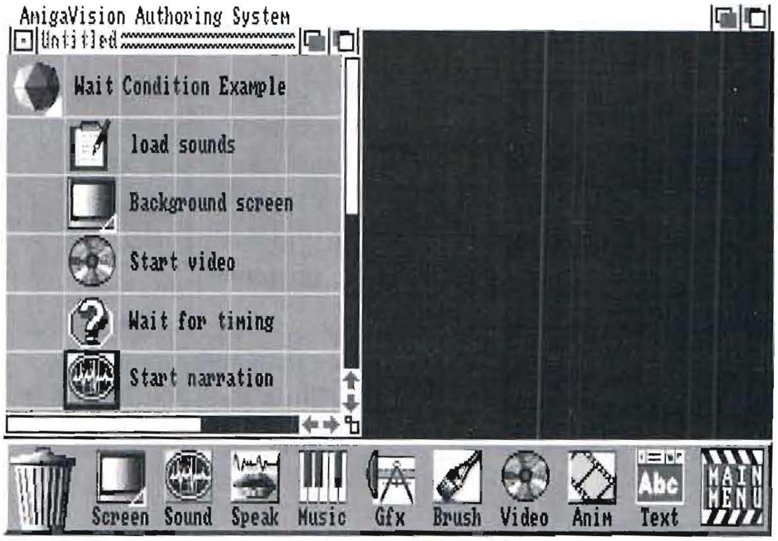
The Wait Condition icon cannot contain children.

Usage

You may use the Wait Condition icon to wait until a condition occurs. For example, this icon may be useful when waiting for a timer to exceed a certain specified value.

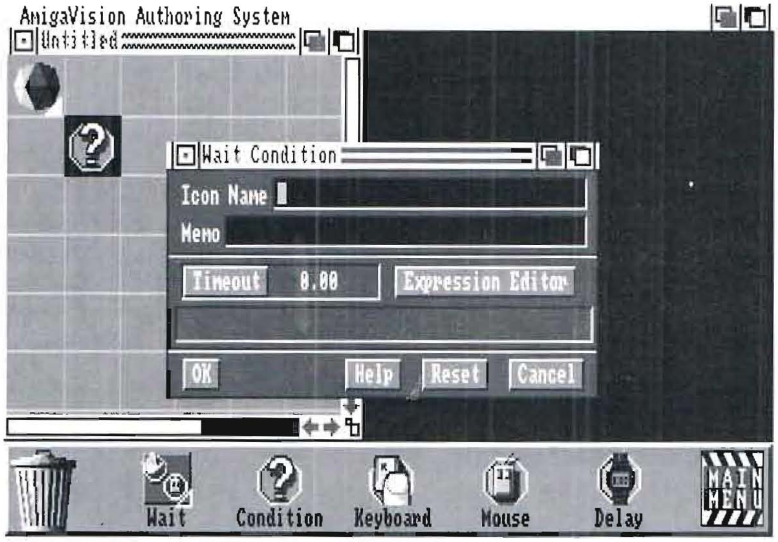
You specify the condition in the Expression Editor.

Wait Condition Example



In this example, digitized sounds are loaded into memory, then a video is started. Sixty seconds into the video, the digitized sound begins to play.

Wait Condition Requester

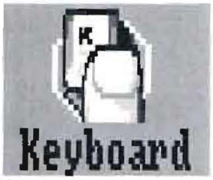


Click on the *Timeout* field to specify the amount of time in seconds the presentation will wait in the event the desired response does not occur. This will open the Specify Value requester. Note that if the Timeout is set to 0 (the default), the program waits infinitely. If Timeout is greater than 0, the program waits the specified number of seconds.

Click on the *Expression Editor* gadget to enter the Expression Editor. This is where you will specify the conditional expression. The text box under the *Editor* gadget will display the expression you create.

You enter the conditional expression through the Expression Editor. The condition uses the timer() function. The parentheses following the timer function enclose the timer number (in this case timer(1)). When the timer(1) is greater than or equal to the value specified on the right hand side of the expression, this condition will be true.

Wait Keyboard Icon



Purpose

The Wait Keyboard icon is used to pause the application and wait for a desired keystroke.

Relation

The Wait Keyboard icon cannot contain children.

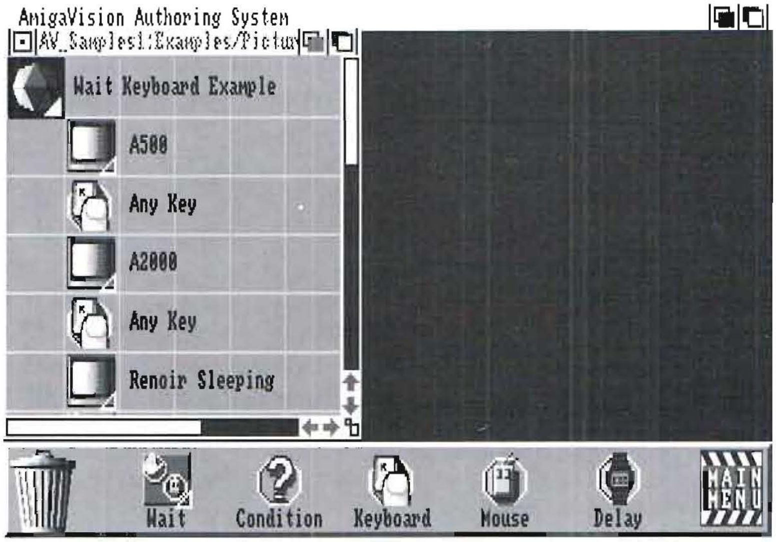
Usage

There are two options when choosing a wait key. The first is to wait for a specific key or keys to be pressed (including function keys, Help and ESC). Second, you may allow for the program to wait for any key to be pressed.

You create display objects and text for the Wait in the Object Editor. The Object Editor is described in Chapter 4.

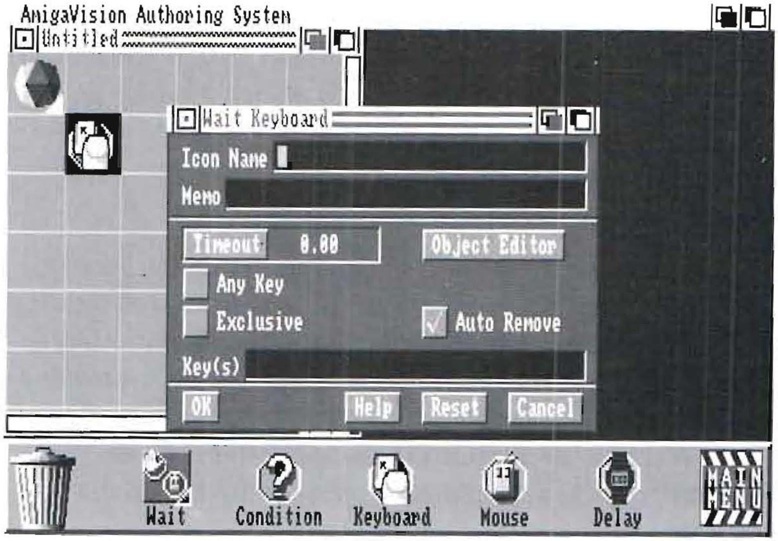
During presentation, the application will pause when it reaches the Wait Keyboard icon. After the desired key press, the application continues with the next sibling icon.

Wait Keyboard Example



The Wait Keyboard waits for a response of any key from the keyboard. When one desired response is given, the program continues with the next icon.

Wait Keyboard Requester


AmigaVision Authoring System

Untitled

Wait Keyboard

Icon Name

Memo

Timeout 0.00

□ Any Key

□ Exclusive

☑ Auto Remove

Key(s)

Click on the *Timeout* field to specify the amount of time the presentation will wait in case the desired response does not occur. This will open the Specify Value requester. Note that if the Timeout is set to 0 (the default], the program waits infinitely. If the Timeout is greater than 0, the program waits the specified number of seconds.

Click on the *Keys* field and type in the specific key or keys to be pressed. If you want to allow more than one key as possible alternatives, separate them by commas. For example, if you want the user to choose from a menu of five items by pressing one of the five numbers, then the key field would appear as: 1,2,3,4,5. For characters the user input is case sensitive; i.e., you can specify the characters "A" and "a" as two separate keys. To wait for the Help, Esc or function keys, spell them out as they appear on the keyboard.

The key pressed by the user is stored and can be checked using the response() function in the Expression Editor (see Chapter 4). For instance, if after a keyboard input you want to branch using an If-Then-Else icon, use the response() function. If you double-click the If-Then-Else icon you will be presented the Expression Editor in which you can enter the following expression:

response() = = "A" or response() = = "a"

There is an on/off gadget next to the *Any Key* option. The *Any Key* option, when set to on, causes the presentation to wait for the user to press any key on the key board before continuing on to the next icon in the flow. For example, you want to provide a feature that pauses an application so that a user can take a break. When the user returns to the application he can continue with the application by pressing any key on the keyboard.

There is an on/off gadget to set *Exclusive* state. *Exclusive ON state* means that only the keys specified by this icon will be active. Other keys that may have been defined previously will not be active during this wait. When *Exclusive* is set to *off* then previously defined hit boxes, keyboard interrupts, etc. will all be active. The user can choose to select any of them. If you want to have interrupts active throughout the application do not set Exclusive to on state.

Click on the *Object Editor* gadget to enter the Object Editor. Here you may create text and other display objects for placement on the screen during the wait. For instance, using the objects you might want to display a text string informing the user "Press Any Key to Continue."

You can set the *Auto Remove* gadget to on or off state. If it is on, then the objects and text (created in the Object Editor) associated with the Wait Key icon will disappear when the desired response is given by the user. If it is off, the objects and text remain on the screen until the action of all its siblings are completed.

Wait Mouse Icon

Purpose

The Wait Mouse icon is used to pause an application and wait for a desired click of the left button on the mouse.

Relation

The Wait Mouse icon cannot contain children.

Usage

You have two options with the Wait Mouse icon. The first is to wait for a mouse click in a specific hit box. Second, you may wait for any mouse click.

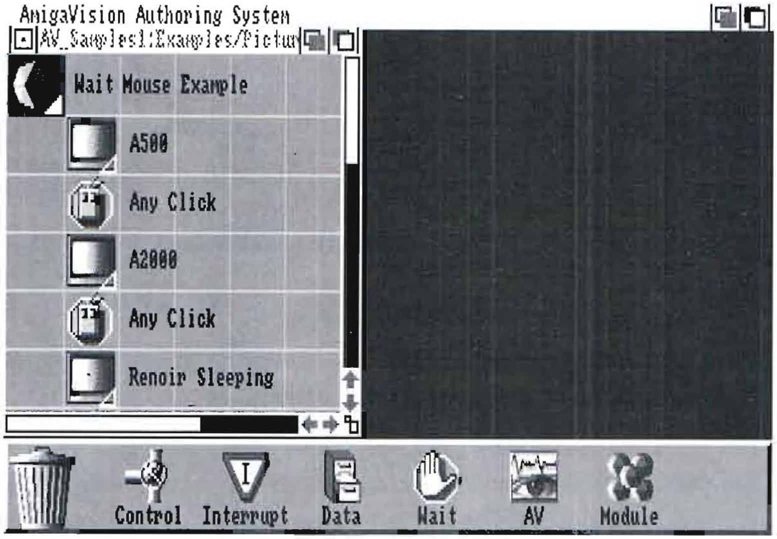
You create objects, text and hit boxes for the Wait Mouse icon in the Object Editor. The Hit Boxes are display objects with a response string specified. These objects can be brushes, rectangles, polygons, circles, ellipses and text strings. You can specify visual and digitized sound actions as feedback to the user.

The response strings specified for hit boxes that are associated with a Wait Mouse icon can be tested using the conditional branching icons (If-Then, If-Then-Else, or Conditional Goto). The response() function in the Expression Editor returns the string that corresponds to the hit box that was clicked on by the user.

The Wait Mouse icon places the display objects and text on the screen. It is a good practice to create the background screen or graphics using the Video, Screen or Brush icons and then use the Wait Mouse icon in the flow to display the objects.

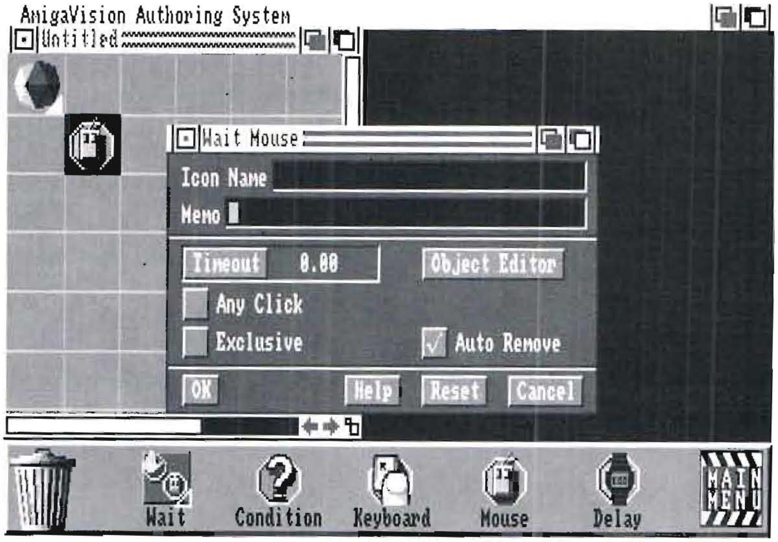
During presentation, the application will pause when it reaches the Wait Mouse icon. After the desired mouse click, the application continues with the next sibling.

Wait Mouse Example



Wait Mouse waits for any mouse click. When the response occurs, the application continues with the next icon.

Wait Mouse Requester



Icon Name
Memo
Timeout 0.00
□ Any Click
□ Exclusive
☑ Auto Remove

Click on the *Timeout* field to specify the amount of time in seconds the presentation will wait in case the desired response does not occur. This will open the Specify Value requester. Note that if the Timeout is set to 0 (the default), the program waits for an infinite time. If the Timeout is greater than 0, the program waits the specified number of seconds.

When the *Any Click* state is *on* it causes the presentation to wait for a mouse click anywhere on the screen before continuing. All mouse clicks are assumed to be the click of the left mouse button. During presentation the right button on the mouse is not checked for by AmigaVision.

When the *Exclusive* state is *on,* it means that only the hit boxes specified by this icon will be active. All the other hit boxes, including interrupts, that may have been defined previously will not be active during this wait. Thus if you want your interrupts to be active, set the *Exclusive* gadget to *OFF* (unchecked) state.

When the *Auto Remove* gadget is set to the *ON* (checked) state, the objects and text associated with the Wait Mouse icon (created in the Object Editor) will disappear when the desired response is given by the user. If it is not selected, the objects and text remain on the screen until the actions of all its siblings are completed.

Click on the *Object Editor* gadget to enter the Object Editor. Here you may place text or objects on the screen to create the appearance of the wait and define the hit boxes and response strings.

You create text hit boxes for the menu in the Object Editor using the Add Text options. You must type a response string in the requester to use the text object as a hit box.

Delay Icon

Purpose

The Delay icon is used to pause the application for a specified number of seconds. It does not require a response from the user.

Relation

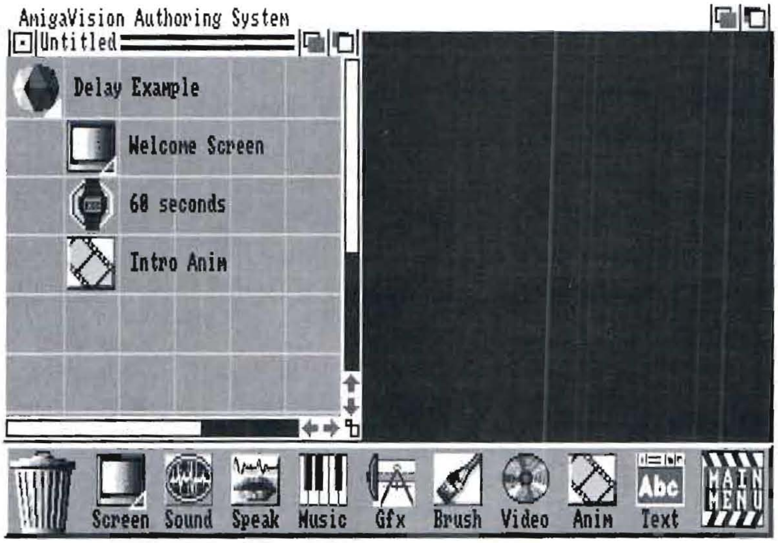
The Delay icon cannot contain children.

Usage

The Delay icon will pause the presentation for the specified number of seconds. When the time has elapsed, the application will continue with the next sibling icon.

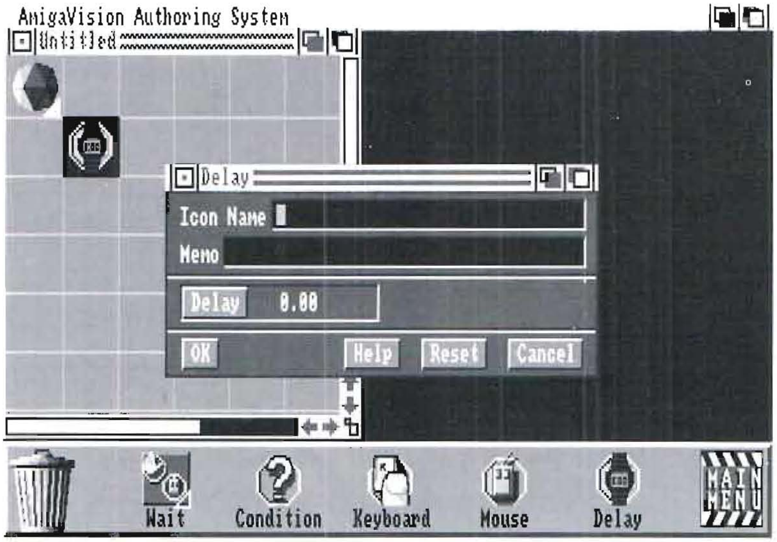
For example, you may want to use the Delay icon after a Brush or a Graphics icon. The Delay will cause the picture or screen to be displayed for the specified amount of time. The screen display will not change until the time has elapsed.

Delay Example



The Delay icon waits for the specified time. In this example, it pauses for 60 seconds to allow the user to read the text before running an animation.

Delay Requester



Icon Name

Memo

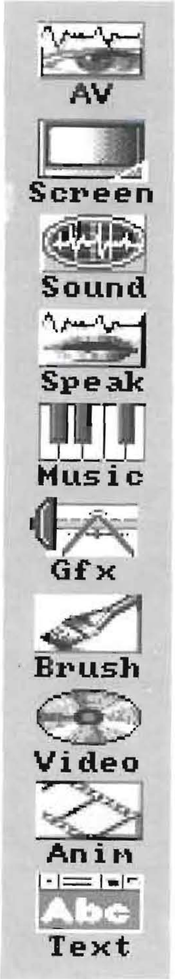
Delay 0.00

Click on the *Delay* field to specify the number of seconds to wait. This will open the Specify Value Requester (Chapter 3).

Audio Visual Icons

Purpose

The Audio Visual icons perform operations such as playing video, animation, sound, speech or music files, and displaying pictures and graphics. You use these icons if you want to display anything on the screen or play back sound or music from either a video device or from files stored on a computer disk.

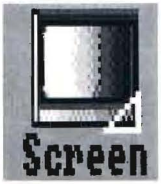


Usage

You must use software other than AmigaVision to create files for picture, music, animation, text, speech and sound. The Object Editor graphics objects are the only audio visual components that can be created within AmigaVision.

In all Audio Visual icons where external files are referenced (such as picture, sound, etc.) you have the option of either directly specifying the filenames or using a variable to specify filenames indirectly.

There is a close relationship between the Content Window in AmigaVision and the Audio Visual icons. You may use the Content window for storing audio and visual files in a library or archive. If you plan to use the same set of audio and visual elements in multiple applications, then it is a good idea to keep these elements organized through the Content window. It is a good practice to organize all audio visual information in a content window, just as you would organize articles and clippings in file folders.



Screen Icon

Purpose

The Screen icon is used to define the background screen for presenting any visual information such as pictures. You use it to specify parameters such as the screen resolution, number of colors, palette, and the size of the picture. You can do this by either loading a picture or by specifying explicit screen parameters.

The Screen icon is also intended for use as a parent to other Audio Visual icons.

Relation

The Screen icon can contain itself and other Audio Visual icons as children. It cannot contain non-Audio Visual icons as children.

Usage

You use the Screen icon to define screen characteristics including resolution, number of colors, and palette. It serves several purposes and can be used in different ways.

You must use this icon in the beginning of an application to define the default screen attributes. You can also use it to alter screen attributes in the middle of an application.

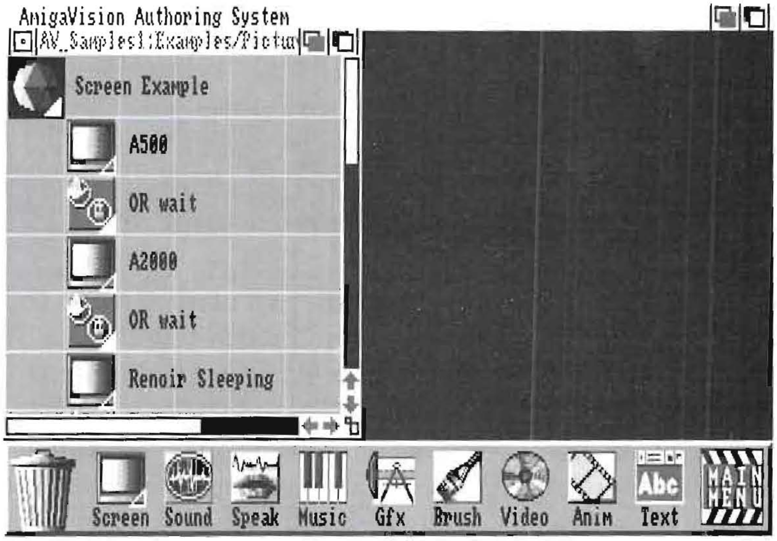
In general, however, it is a good practice to use the same screen resolution, colors, and palette throughout an application so as not to disorient the viewer.

When you use the Screen icon to load and display a picture, the picture will erase the current background. If you do not want the existing background to be erased use the Brush icon instead.

You may change the palette of the picture you are loading, as well as the screen resolution. You may also specify where on the screen the picture will be displayed using screen pixel coordinates.

The Screen icon can also be used to clear the screen by placing the screen icon in the outline, and leaving the filename field blank.

Screen Example


AmigaVision Authoring System

AV_Samples1:Examples/Pictu

Screen Example

A500

OR wait

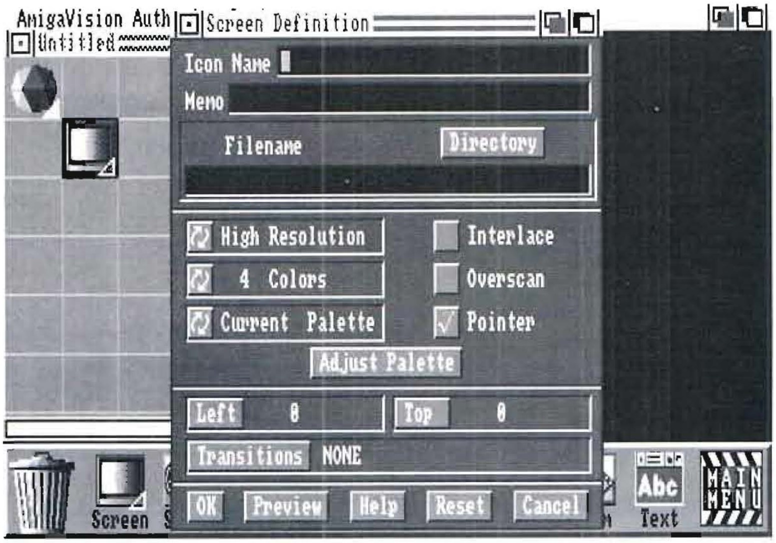
A2000

OR wait

Renoir Sleeping

In this example the first Screen icon loads a picture onto the screen. It wipes out any previous pictures or graphics on the screen. It then waits for a key press or mouse click. After the response the second screen icon erases the screen and displays a new picture.

Screen Requester



Screen Definition

Icon Name

Memo

Filename

High Resolution

4 Colors

Current Palette

□ Interlace

□ Overscan

☑ Pointer

Left 0

Top 0

Transitions NONE

To specify the name of the picture file to retrieve, click on the Filename field and type in the name. You may also click on *Directory* to open the File requester. You may use this file simply as a reference for setting the background or actually loading it. As soon as you type the filename in the field (or alternatively select from the file requester) and press the Enter key, all the screen parameters such as the resolution are read from the specified picture file. However, if you do not want the picture displayed but want to retain the screen parameters, then click again on the Filename field and use the Del key (or Right Amiga-X to clear the whole line) on your keyboard to erase the filename.

You can also use variables to specify filenames as described in the File Requesters section of Chapter 3.

The filename can refer to either a picture, a brush, or an animation file. If you choose an animation file the Screen icon will display the first frame of the animation.

Under Screen Definition, there are several modes available. Click on the first Multistate gadget on the left to select between *High Resolution, Extra Halfbrite, Hold and Modify, Current, File Defined* or *Low Resolution*. The Current option simply retains the current screen as it was defined by a previous Screen icon in the flow of an application. If you use a variable for filename and you want to use the screen definition from the IFF file at presentation time, then you should choose the option called *File Defined*.

The next Multistate gadget refers to the number of colors used. Depending on what you chose for screen resolution, there are restrictions on the available number of colors. You can specify your selection using the color gadget.

The third Multistate gadget on the left selects the palette as either *Original, Current* or *Modified*. Current refers to the palette used in the application flow just before the Screen icon. Override tells the screen icon to ignore the previous palette and use the specified one (after using Adjust Palette, below). The Original setting causes AmigaVision to use the palette defined in the file when the application is run using Present.

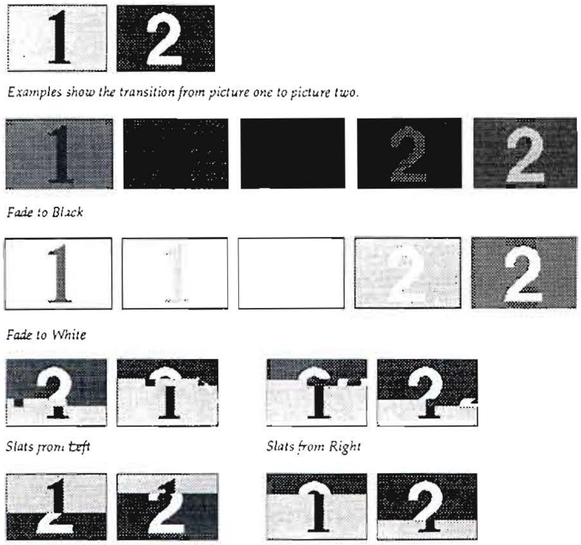
Each of the screen modes may be in either *Interlace* and/or *Overscan*. To select either of these options, click on the on/off gadget on the right side of the requester, and a check mark will appear to indicate your selection. The third on/off gadget, Cursor, is to specify whether you wish to have the mouse pointer appear on the screen or have it turned off. If the check mark is in On state (default), the mouse pointer will be displayed.

Choose the *Adjust Palette* gadget if you wish to make changes to the palette from a specified picture file. The Palette requester is presented. First click the mouse pointer on a color of your choice and the slider gadgets will show the current settings for that color. Use the slider gadgets to adjust that palette color.

Left and Top gadgets are used to specify the coordinates of the top left corner of the picture. Click on either gadget to enter the Specify Value requester. Here you can specify the top and left coordinates of the screen in terms of pixels in current resolution. Note that 0,0 refers to the top left corner of the screen. You are allowed to specify negative values of top or left coordinates if you want to display only part of a picture.

Click on the *Transitions* gadget to specify the visual effect to be used when switching pictures. The Transitions requester is presented. There are a number of screen transition effects, dissolves, slats and fades, available as options. It is important to note that you cannot use any of the screen transition effects except Fades when you are going from one screen resolution to another. You can also use the Fades to create transitions from one screen color to another.

Transition Examples



Examples show the transition from picture one to picture two.

Fade to Black

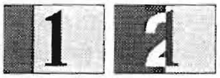
Fade to White

Slats from Left

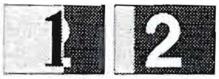
Slats from Right

Wipe from Bottom

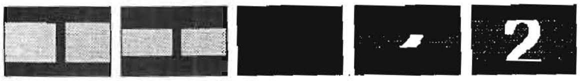
Wipe from Top



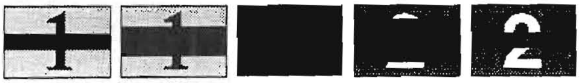
Wipe from Left



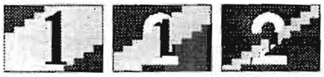
Wipe from Right



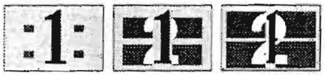
Horizontal Unfold



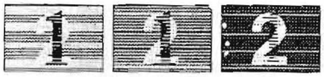
Horizontal Fold



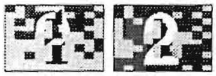
Diagonal Contract



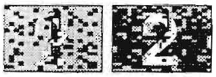
Expanding Quadrants



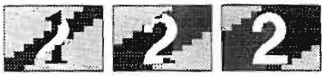
Blinds



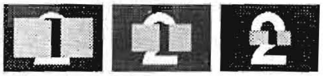
Big Dissolve



Small Dissolve



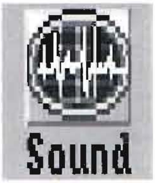
Diagonal Expand



Collapse to Center

Also if you are using transition effects when going from one palette to another, the old palette is used until the transition is completed, and then the screen colors are set to the new palette.

For these reasons it is recommended that you use a single resolution and palette throughout an application, if you want to use transition effects. Experiment with them until you find a satisfactory transition for your application. It is a good practice to choose one or two effects to use throughout the application to maintain continuity and avoid disorienting the viewer.



Digitized Sound Icon

Purpose

The Digitized Sound icon is used to play a recorded voice or sound that has been previously digitized.

Relation

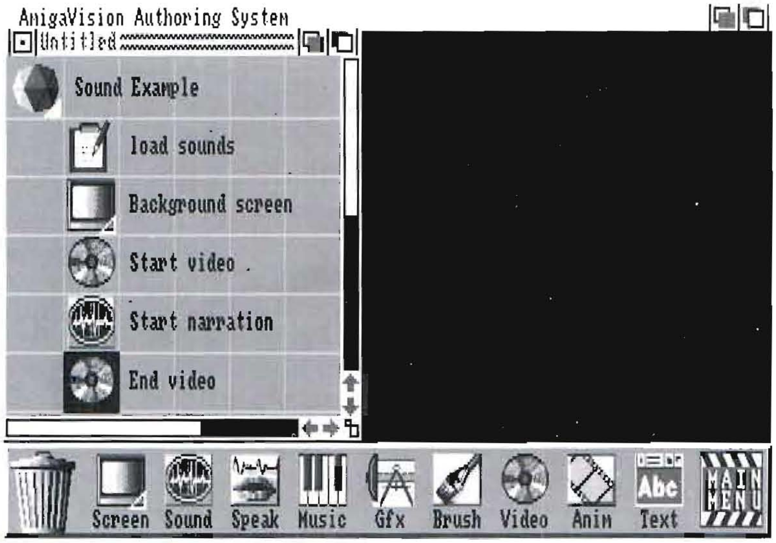
The Digitized Sound icon cannot contain children.

Usage

You can use this icon to play back a digitized sound file.

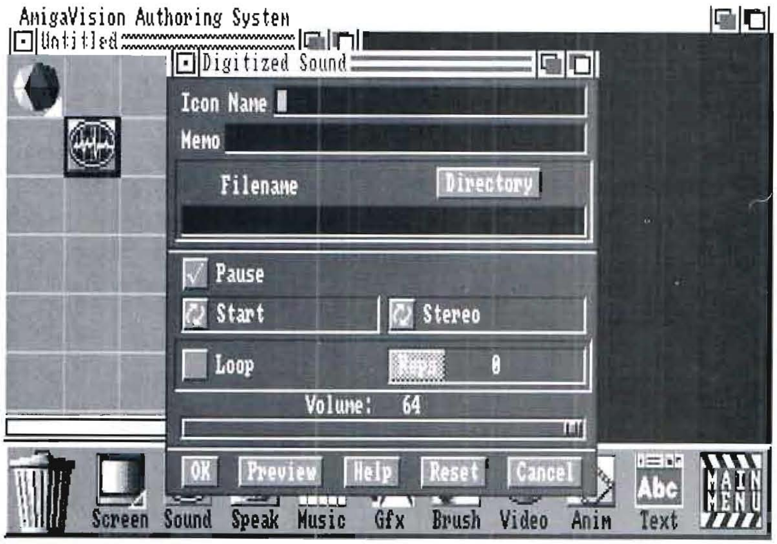
Note that the digitized sound may be any IFF sound file, either an instrument (multi-octave) or 8SVX (one-shot) file. Most commercially available sound digitizing programs offer the option of creating the file in this format.

Digitized Sound Example



In the example, sounds are first loaded into memory. After the video begins, the digitized narration plays.

Digitized Sound Requester



Icon Name

Memo

Filename

☑ Pause

Start

Stereo

□ Loop

0

Volume: 64

To specify the sound file to be used, click on the Filename field and type in the name. Or you may click on the Directory gadget to open the file requester.

You can also use variables to specify filenames as described in Chapter 4.

Using the *ON/OFF* gadget select Pause *ON* or *OFF*. *Pause ON* means halt the application until the sound file has been played back completely. This is the default option. The application moves to the next icon following the sound icon after the completion of the sound event.

The *Pause OFF* state starts the sound and then continues to execute the next icon in the outline. Turn the Pause off if you want to start the sound and immediately also start executing the action of the following icon such as an animation or a video sequence. This way you can combine sound and video.

The *Speaker* gadget allows specification of Left, Right or Stereo sound. Stereo samples will be played back in stereo; the stereo effect will be simulated if mono samples are played back as stereo.

You can simultaneously play a number of sound files. There are four channels in Amiga hardware and each mono sound file takes up one channel to play back. Each stereo file uses two channels. You can have one sound playing through the left speaker, another through the right, and have a third stereo sound play back on both. You can use this feature to superimpose multiple sounds to create special effects.

Click on the Multistate *Sound* gadget to specify *Start*, *Stop* or *Stop All*. The *Start* and *Stop* gadget requires a specific sound filename to either start or stop playing the sound. The Stop All option, which does not require a filename, stops all the sounds that are being played back simultaneously.

You can click on the *Loop* and the *Rep* gadgets to access the Specify Value Requester. In this requester you can enter the number of times you want the specified sound to be repeated. The number 0 will cause the sound to repeat endlessly.

Finally, you can use the volume control slider gadgets to adjust the volume. Preview the sound to make sure the volume is consistent with volumes of other sounds in the application. The range of index for volume selection is from 0 to 64. This setting overrides the volume settings in the Defaults (see Chapter 4 under the Project Menu for details on Defaults).

Synthesized Speech Icon

Purpose

You can use the Speech icon to play back keyboard text that you input, or an ASCII text file using Amiga's built-in speech synthesis capabilities.

Relation

The Synthesized Speech icon cannot contain children.

Usage

The Synthesized Speech icon can be used to play back a text string or a text file. It allows for synthesized speech to be played along with motion video or stationary picture backgrounds.

If a text file is used, it must be in ASCII format. To create an ASCII file refer to the manual for the word processor you are using.

Alternatively, you can type a text string directly in the text field if your message is 255 characters or less.

To create synthetic effects such as different voices, you may specify the rate, pitch, mode, sex, and volume.

You may specify these parameters in the requester as explained later or you may include them in the word processor file. To include them in the file, use this structure anywhere in your text file:

|Rate, Pitch, Mode, Sex, Volume|

The available values are:

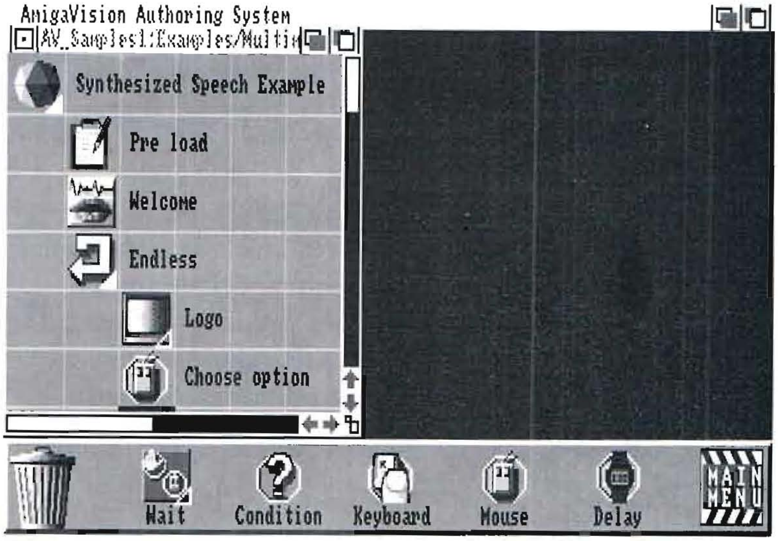
Rate	40 – 400
Pitch	65 – 320
Mode	0 – 1 (Where 0 is natural and 1 is robotic)
Sex	M – F
Volume	0 – 64

For instance, if you want to simulate a conversation between a male and a female, then your text file might appear as:

|100,120,0,0,64|
Stacy, did you see my sneakers?
|100,300,0,1,64|
No, Jim. You must have lost them.

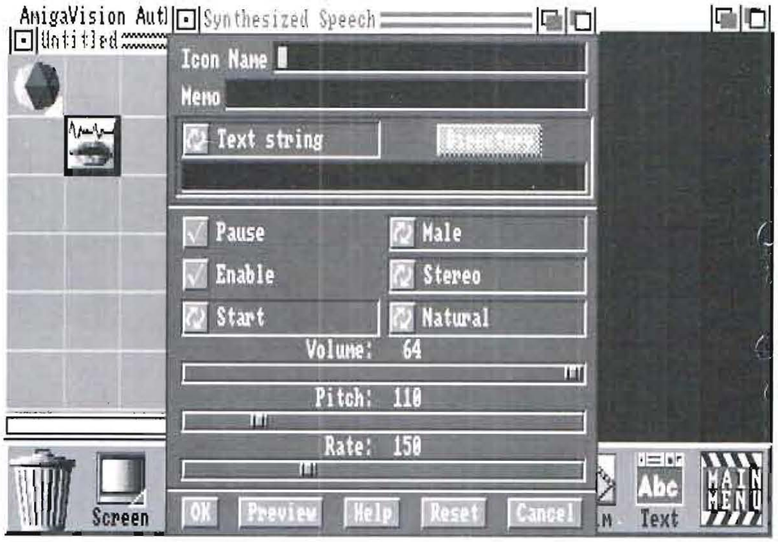
The vertical bar at the beginning and end of the parameters is a character on your keyboard that is just one key to the left of the backspace key. (SHIFT-\)

Synthesized Speech Example


AmigaVision Authoring System
AV_Samples1:Examples/Multi
Synthesized Speech Example
Pre load
Welcome
Endless
Logo
Choose option
Wait
Condition
Keyboard
Mouse
Delay
MAIN MENU

In this example, the Synthesized Speech is played before the loop begins.

Synthesized Speech Requester



Icon Name
Memo
Text string
☑ Pause
☑ Enable
Start
Male
Stereo
Natural
Volume: 64
Pitch: 110
Rate: 150

If you want to specify a filename, first use the Multistate gadget below the Memo field to select *Filename* as the option. Then use the *Directory* gadget to open the File requester (Chapter 3) where you may specify the volume, drawer, and name of the file to be used. You may also click on the field below the *Directory* gadget and type in the desired filename.

You can also specify the filename as a variable as described in Chapter 4.

If you are entering a text string, type the desired text in the field; be sure that the Multistate gadget above the field is set to *Text String*. You can enter a maximum of 255 characters for text string; for longer text you must use a file. You may also enclose variable names in brackets for Speech to say. For example, if a variable is defined using the Variables icon as Number = "One" and you type in the Text field:

The number is [Number]

then the synthesized voice will say "The number is One" during play back.

On the left side of the requester there are three gadgets. The first one is to turn *Pause* On or Off. The default state is On. If *Pause* is On the speech is played back completely before the application moves on to the next icon in the flow. If *Pause* is Off then the speech is started and the application immediately proceeds to execute the action of the following icon.

The next gadget below the Pause sets *Enable* state to On or Off. With Enable set on, you can use all the settings gadgets to define the voice parameters. If Enable is set off, then the settings gadgets are ignored and void parameters are taken from the previous Speak icon.

The Multistate gadget below the *Enable* gadget is to specify Start or Stop of playback of speech.

To set voice parameters in the speech requester you may use the three Multistate gadgets on the right hand side. The top gadget lets you choose the Sex as *Male* or *Female;* the next gadget below lets you select the speaker (i.e., *Left, Right,* or *Stereo*); the next one below lets you choose between *Robotic* or *Natural* voice.

You can set the *Volume* using the slider gadget. The range of volume index is from 0 for the lowest to 64 for the highest volume. This setting overrides the value specified in the Defaults option.

Pitch refers to the tone of voice used. Use the slider gadget to set the desired pitch. The lowest index value for setting is 65 and the highest is 320.

Rate refers to the speed of the speech. Use the slider gadget to set the desired rate. The lowest index value of speed is 40 and the highest is 400.

Music Icon

Purpose

You can use the Music icon to play back musical scores created in music software programs.

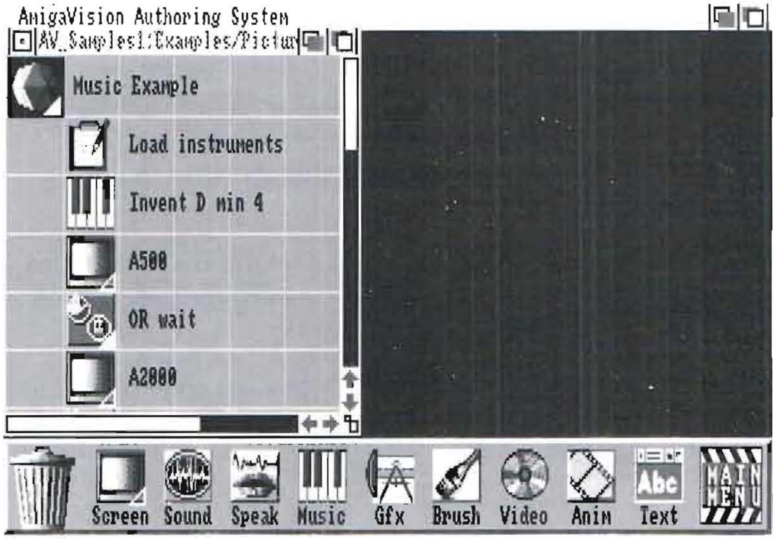
Relation

The Music'icon cannot contain children.

Usage

The Music icon may be used to play a music file by itself or along with video, animation or graphics. The file to be played must be in SMUS format. Files in SMUS format can be created using the Deluxe Music Construction Set (published by Electronic Arts) or using other software with a similar file format. Playing SMUS files will use up all four Amiga sound channels, so you cannot play sound files at the same time.

Music Example

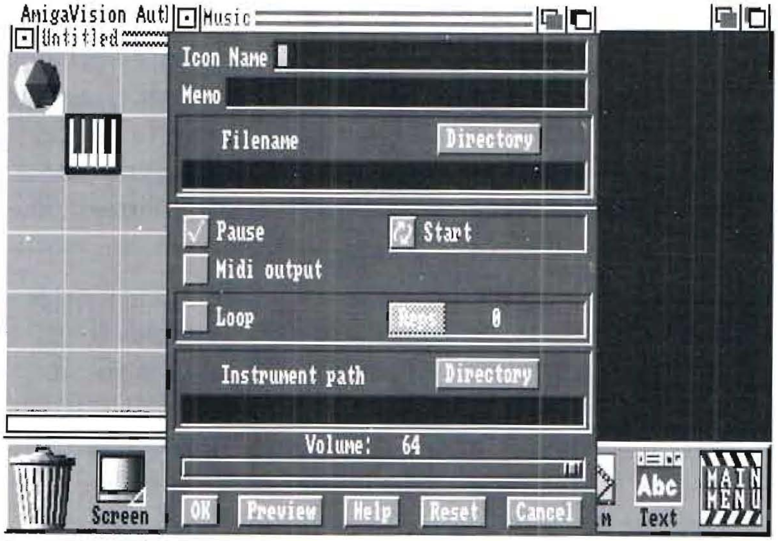

AmigaVision Authoring System

AV_Samples1:Examples/Pictur

Music Example
- Load instruments
- Invent D min 4
- A500
- OR wait
- A2000

In this example, the Music icon plays a music file along with the display of screens.

Music Requester



Icon Name
Memo
Filename [Directory]
☑ Pause
Start
□ Midi output
□ Loop 0
Instrument path [Directory]
Volume: 64

The *Directory* gadget opens the File requester where you specify the name of the music file to be played. You may also click on the Filename field and type in the desired filename.

You can also specify the filename as a variable as described in Chapter 4.

The *Pause ON* state is set as default, and it is useful if you want the project to wait until the music file has completed playing before performing the action of the next icon. If you want the icon following the music icon to execute immediately, then set Pause state to Off.

The *MIDI Output* is set to *On* if you want the music to be played through a MIDI instrument. A maximum of four MIDI channels are supported. Note: You must have a midi interface hooked up to your Amiga's serial port, and the proper midi cables connected from the interface to your keyboard. As a rule MIDI "outs" connect to MIDI "ins" and vice-versa. Please consult the documentation that came with your keyboard and MIDI interface before making any connections.

Click on the Multistate *Start/Stop* gadget to specify whether you want to start or stop the music file. It may be necessary, for example, to stop the music file playback, if a user interrupts the music to ask for help. Once stopped, however, you cannot resume at that point after the completion of an interrupt. You have to start again at the beginning of the music file.

You must specify an instrument path in the text field provided for this purpose. A music score file will not play back if AmigaVision cannot find the path for instruments in your disk. Instruments refer to digitized sounds from various instruments such as piano, cello, horn, etc. that are used in your score file. Please refer to your music software manual for more information on instruments. Most music software available allows you to sample your own sounds and create custom instruments.

You can also specify an instrument path using the Defaults option listed under Project in the pull-down menu in the flow editor. If you have specified a default path it will be displayed in the field provided for the path.

You can play back the music file repeatedly a number of times using the *Loop* feature. If you set the Loop state to on, the *Reps* gadget will be activated and you can click on it to specify the loop count. The number 0 will cause the music to repeat endlessly.

You can use the slider gadget to control the volume of the music. The range for the index value for volume control is 0 to 64.

Graphics Icon

Purpose

The Graphics icon is used to modify and control the screen display. It enables you to place display objects (created in the Object Editor) on the screen. It also lets you specify color cycling effects.

Relation

The Graphics icon cannot contain children.

Usage

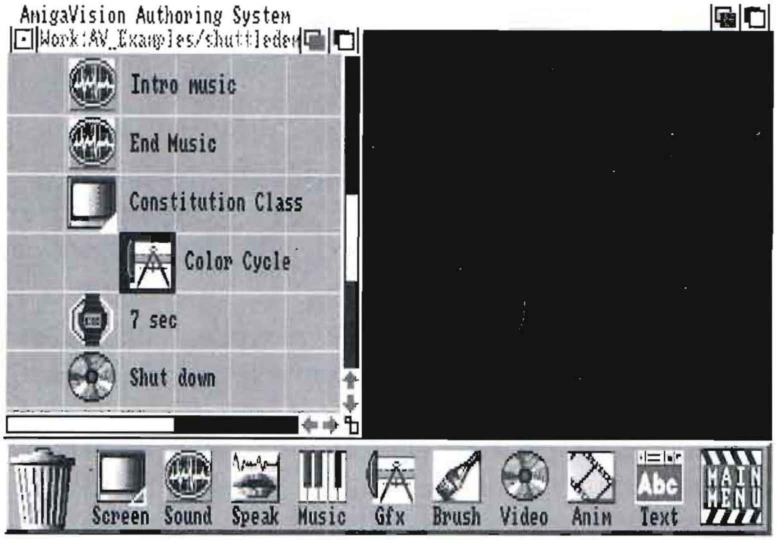
You may use the Graphics icon to display text and objects which have been created in the Object Editor. Graphic objects can include: brushes, text, circles, rectangles, lines, ellipses, or any arbitrarily polygons.

You may also use the Graphics icon to turn color cycling on or off. The color cycling is a very powerful way to create simple animation effects such as the rotating of gears in a machine or the flow of water. The use of this feature requires that you have saved your pictures with color cycling effects specified using your compatible paint software. Please refer to your paint program's software manual for more details on how to create pictures with color cycling effects.

Color cycling requires three parameters that you specify in your paint package. The first is the number of ranges of colors that will be cycled. DeluxePaint III, for instance, supports up to six separate ranges. AmigaVision supports a maximum of four. The second parameter is the speed of cycling. The third parameter is whether the cycling of colors should be in the forward or the reverse direction.

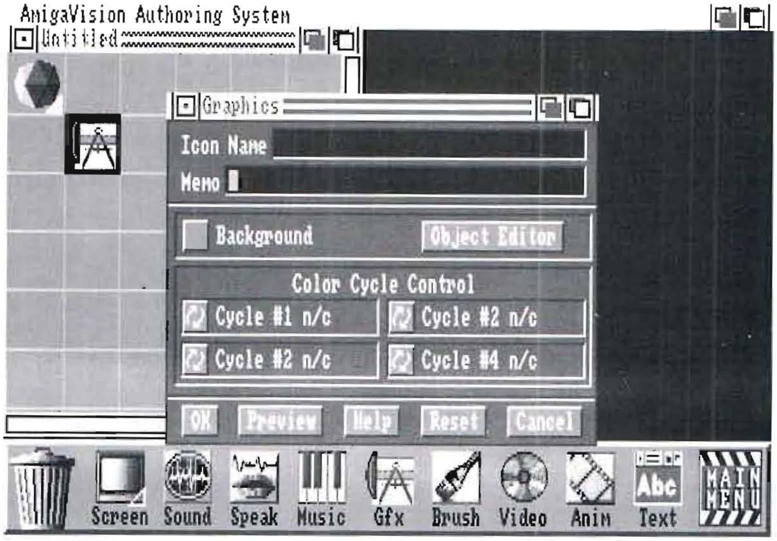
Remember that the Graphics icon will only display the text and objects you create in the Object Editor. The background picture that you use in the Object Editor is not displayed during presentation. If you wish to display the picture and display objects on it, use the Screen or Brush icon to place the picture on the screen before using the Graphics icon.

Graphics Icon Example



In this example a picture is color cycled using a Graphics Icon.

Graphics Requester



If you click on the *Background* gadget, the objects are stamped into the background. This means that the objects will remain on the screen until the background is replaced (i.e., by a Screen icon).

If Background is not selected, objects will only remain on the screen until the actions of this icon and its siblings have been performed.

Click on the *Object Editor* gadget to enter the Object Editor where you will create the text and objects to be used on the screen.

There are four separate Multistate gadgets under the heading of Color Cycle Control. Each of these refers to one of the color cycling ranges that was saved in the picture (IFF) file. You can control each of the ranges.

Each of the Multistate gadgets has four possible states. You can start the color cycling in the *Forward* or the *Reverse* direction; turn the cycling effect *Off*, or leave it unchanged *(N/C)* if it has been turned on by a previous Graphics icon.

Brush Icon

Purpose

The Brush icon is used to overlay a specific IFF picture file or brush (miniture picture) on top of the current screen background.

Relation

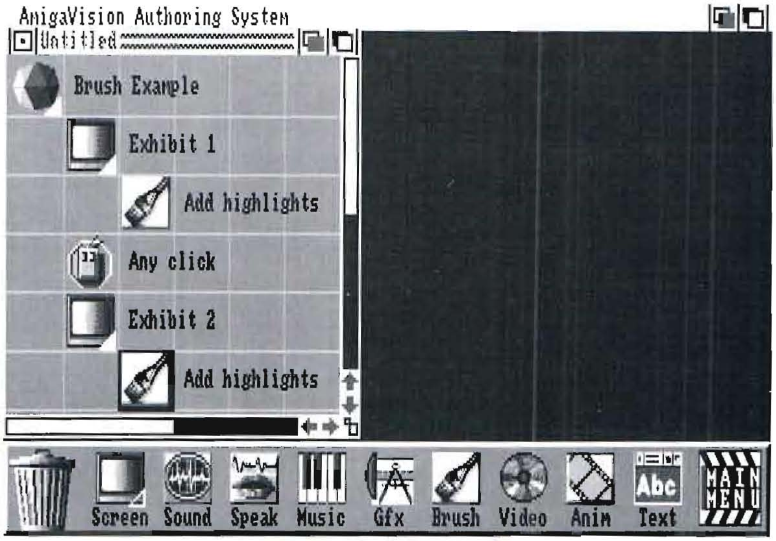
The Brush icon cannot contain children.

Usage

The Brush icon displays an image on the screen. It differs from the Screen icon in that it does not delete the existing background pictures or graphics on the screen. It also does not modify screen attributes such as resolution and colors.

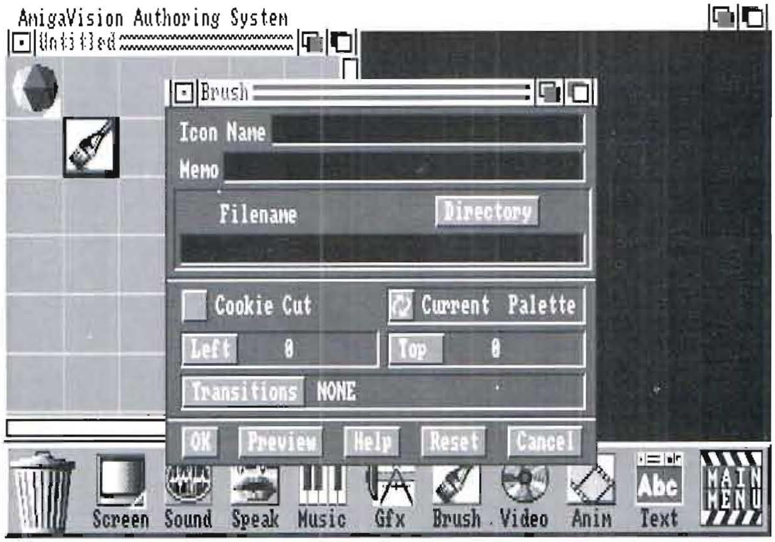
Therefore, if you want your brush displayed normally, your brush file should be in the same screen resolution as the background picture file.

Brush Icon Example



The Brush icon adds highlight brushes to a picture file. The Wait Mouse waits for any click. When the mouse is clicked, the Screen icon clears the screen.

Brush Requester



Icon Name
Memo
Filename
□ Cookie Cut
Current Palette
Left 0
Top 0
Transitions NONE

The *Directory* gadget opens the File requester where you specify the name of the brush to be displayed. You may also click on the Filename field and type in the desired *filename.* The available brush files include standard pictures, brushes and animation. If you specify an animation file, the first frame of the animation will be displayed.

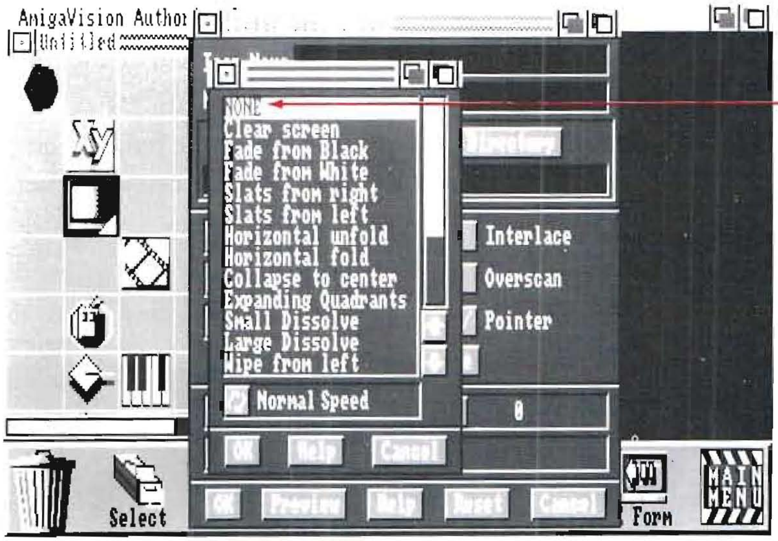
You can also specify the filename as a variable as described in Chapter 5.

When you load an arbitrarily shaped brush on the screen, a rectangle enclosing the brush is shown as the default case. If you do not want this enclosing rectangle shown, then set the *Cookie Cut* gadget to the ON state.

Click on the Multistate palette gadget to specify *Current* or *Override.* If Current is selected, the brush will be displayed using the current background palette. The Override option which is the default, uses the brush palette, changing the existing colors of the background.

Left and *Top* gadgets are used to specify the coordinates of the top left corner of the brush. Click on either gadget to enter the Specify Value requester. The coordinate value of 0,0 refers to the top left hand corner of the screen. You can specify negative values for the coordinates if you want to show just part of the brush.

Click on the *Transitions* gadget to specify the screen pattern to be used when switching pictures. The Transitions requester is presented. All effects except for the Fades are available for transitions within the area of the brush. Fades work on the whole screen rather than on the area of the brush.



transitions requester

Video Icon

Purpose

The Video icon is used to play a segment of video or a single video frame from a videodisc player.

Relation

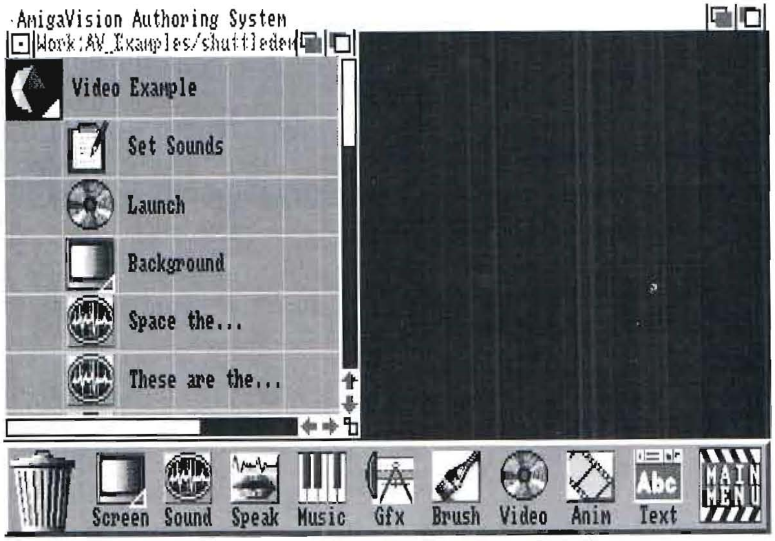
The Video icon cannot contain children.

Usage

The Video icon allows you to display motion video segments or stillframe images. You can overlay the graphic and display objects over video using a genlock. The video segments may be played alone or accompanied by other audio-visual elements such as voice, music, text or graphics.

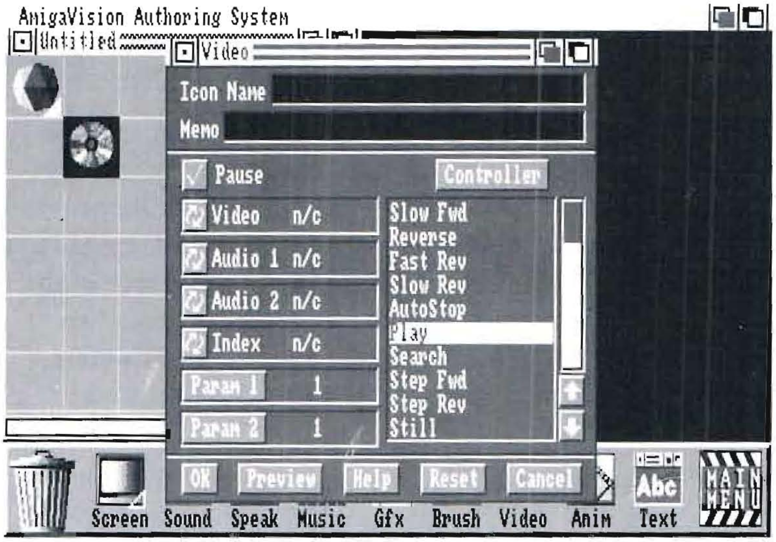
After the video is finished playing, the last frame remains on the screen. You may wish to clear the screen by using another video icon to search to a black video frame.

Video Icon Example



In this example, the Video icon begins a video segment prior to the display of a background screen.

Video Requester



Icon Name

Memo

☑ Pause

Video n/c

Audio 1 n/c

Audio 2 n/c

Index n/c

Param 1 1

Param 2 1

Select *Pause on* to cause the application to wait until the video is finished before executing the next icon in the outline. Select *Pause off* to set the video playing, then move along to execute the next icon, such as a Wait Mouse icon.

There are four Multistate gadgets, representing *Video, Audio Channel 1, Audio Channel 2,* and *Index* display, for specifying the state of the videodisc player. Each gadget has three possible states: *On, Off,* or No Change *(N/C).* For example, if On or Off is selected for Video, the video signal will be turned on or off if the videodisc player has the feature. Some players do not allow video to be turned off. The two audio channels can be controlled similarly to the video. If N/C is selected then the status of the player will remain as is. Index refers to the frame number which appears on the screen as displayed by the videodisc player. AmigaVision supports a number of videodisc players and these are listed in Section 1.

There are two methods by which you can specify the videodisc command and the player status in AmigaVision. One is by selections in the video requester. There is a list of commands shown in the command window. These are the most general commands that are supported by all videodisc players. The second method of specifying the videodisc command is by interactive control of the player using the Video Controller.

Click on the video command of your choice in the window. The command will appear highlighted. Certain commands such as Play, Search and Autostop require specific frame numbers. If you select these commands the Parameter gadgets *(Param 1* and *Param 2)* shown below will let you specify the parameters. For instance, the Play command requires two parameters, i.e., the starting and the ending frame numbers. The Search command requires one parameter—the frame number.

If you know the video frame numbers, you may enter them from the Specify Value Requester which you can access by clicking on the Parameter Gadgets. Alternatively you can assign variable names in which the frame numbers have been stored. The variable names that have been defined previously will be displayed in the Specify Value Requester.

To use the second method of specifying a video command, click on the Controller gadget to enter the Videodisc Controller. The Videodisc Controller is described in Chapter 4. Here you may interactively control the videodisc player and select frame numbers, the video command, and the player state. When you click on the Save gadget in the Controller, everything required for the video requester is transferred automatically. This spares you the effort of noting frame numbers and commands. Any frame numbers selected will be entered into the two parameters. However, you cannot specify variable names using this method.

Animation Icon

Purpose

The Animation icon is used to play back an animation file.

Relation

The Animation icon cannot contain children.

Usage

The Animation icon plays animation files which have been created in paint or animation packages. The animation file must be in ANIM OPT 5 format.

Animation may be used in place of video to display moving pictures. You may also play animation files along with video to combine the effects of live video with animated characters.

You can play back Animation files along with digitized sound, synthesized speech or music by first starting the sound with the Pause off mode and then starting the animation. Alternatively you can first start the animation and then synchronize sound, but you cannot currently specify synchronization to start at a certain animation frame number.

Pause on (default) state will make your application wait until the Animation is finished before executing the next icon.

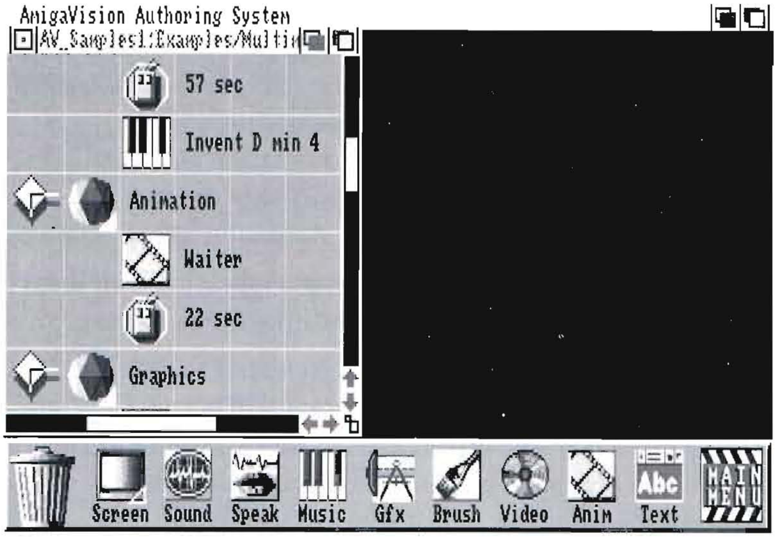
In the ANIM file format the first image specifies the background picture on which animation will occur. Thus you can use the Screen and Brush icons to load an animation file, and the first picture or frame of the animation will be displayed. By using the Screen icon you can displace the previous background, resolution and palette.

If there was an existing background image on the screen before starting an animation, and the override Screen is off, the first frame of the animation becomes part of the background (as if it were a brush). Then the animation starts by displaying changes to the screen. When the animation ends, the last frame again becomes a static part of the background unless the animation sequence is repeated.

You can overlay display objects on a screen with animation in the background as long as the animation does'nt overlap the objects. For instance, you might want to have a hit box with a message "Click Here For Help." When a user clicks on the hit box, you can have an interrupt action execute. The animation will pause until the help is provided by the interrupt, and it will resume again from the beginning.

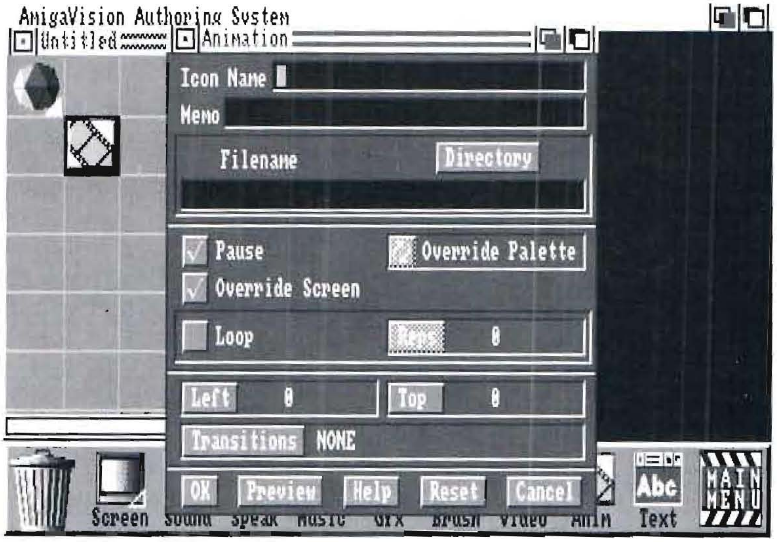
The Screen, Brush and Animation icons can all be used to stop the current animation.

Animation Icon Example



In the example the animation icon is a child of the Module. After 22 seconds the animation exits.

Animation Requester


AmigaVision Authoring System

Untitled

Animation

Icon Name

Memo

Filename

☑ Pause

Override Palette

☑ Override Screen

□ Loop

Reps 0

Left 0

Top 0

Transitions NONE

The *Directory* gadget opens the File requester (Chapter 3) where you specify the name of the animation file to be played. You may also click on the Filename field and type in the desired filename.

The *Override Screen* gadget defaults to ON and uses the screen resolution of the animation file, completely replacing the previous screen. If you click on it to turn it OFF, it will assume the current resolution (i.e., the last screen that was displayed).

Click on the *Palette* gadget to specify Current or Override. If Current is selected, the animation will be displayed using the current palette. If Override is selected, the animation's palette will be used, changing the existing colors.

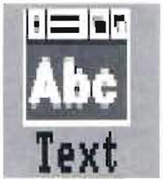
Select *Loop* if you wish to play an animation which has end frames that loop back to the beginning frame (a looped animation). The looping action is specified at the time you create the animation file. Certain animation software like DeluxePaint III will automatically save the animation as a looping type. In other animation software such as Photon Paint, you have to explicitly specify looping. ANIMagic, from Aegis, has a function to take ANIM files created in other products and make them loop endlessly.

If you select Loop, specify the number of repetitions in the Reps field. When you click on the field, the Specify Value requester (Chapter 3) is presented. Note that if you specify zero in the Reps field, an infinite loop is created.

Left and *Top* are used to specify the coordinates of the top left corner of the picture. Click on either field to enter the Specify Value requester. Here you can enter the coordinate values. Note that the Left coordinate has to be an exact multiple of 8 so that the animation can be located on a byte boundary. Even if you enter an inconsistent number such as 28 it will get rounded to 24. The Top coordinate, however, can be any integer within the screen resolution. Negative coordinates are not allowed.

Click on the Transitions gadget to specify the screen pattern to be used when switching to the first picture of the animation. The Transitions requester (Chapter 3) is presented.

Note that Animation will not override screen attributes such as resolution unless Override Screen is ON. The attributes can be changed only by the Screen icon.



Text File Icon

Purpose

The Text File icon is used to display text from an ASCII file onto the screen.

Relation

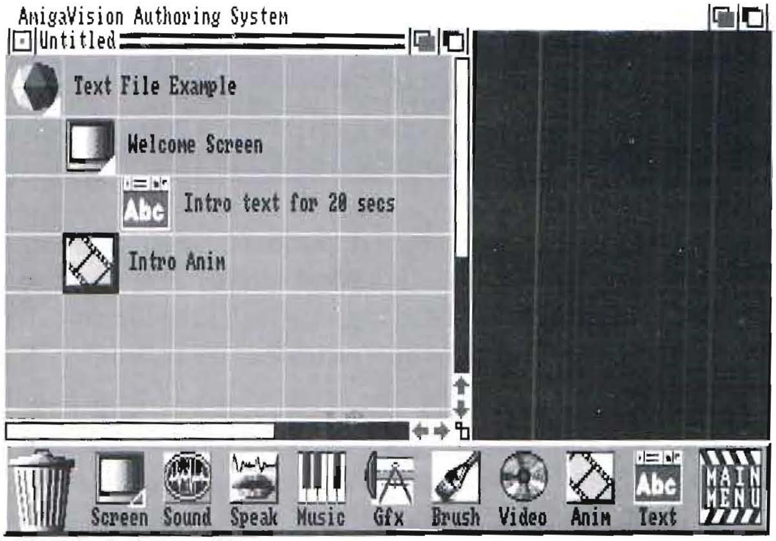
The Text File icon cannot contain children.

Usage

You may use this icon to display text on the screen. The text comes from a file and is displayed in a text window on the screen on top of an existing background. Only one text window can be displayed at a time on the screen. The text file must be in ASCII format.

You must specify the size and the screen location of the text window, the text colors, and highlights using the Object Editor.

Text File Icon Example



AmigaVision Authoring System

Untitled

Text File Example

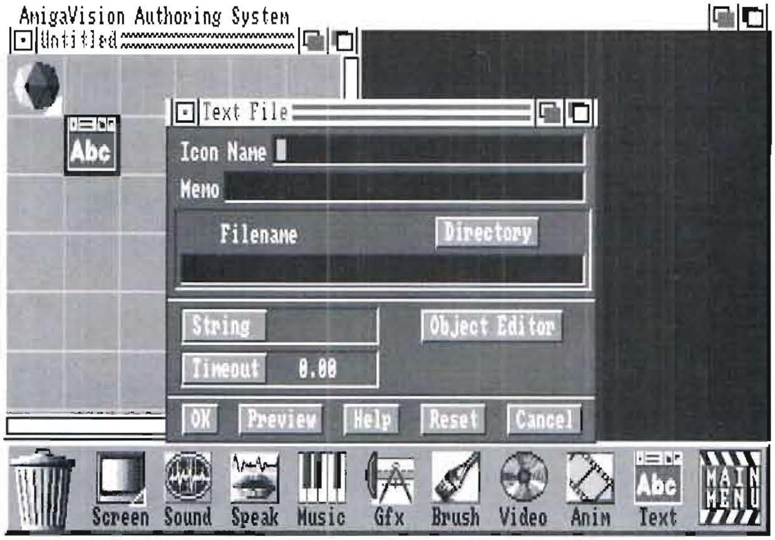
Welcome Screen

Intro text for 20 secs

Intro Anim

In this example the Screen icon loads a picture. Text File places text in a window on top of that picture. The Text File icon allows for specifications of size and coordinates of the box in which the text will be displayed.

Text File Requester



Timeout 0.00

The *Directory* gadget opens the File Requester where you specify the name of the text file to be displayed. You may also click on the Filename field and type in the desired filename. You can specify the filename as a variable as described in Chapter 3.

Click in the String gadget to open the Specify Value requester (Chapter 4). Specify a sting variable containing a specific string value to identify the place in the file at which to begin displaying text. If the first occurrence of the string is found, text is displayed from that point onward. The search is case sensitive so that you can search to specific locations in a text file. This feature is useful in defining a glossary of terms.

You can, for instance, create an ASCII text file with a list of words, each followed by its definition. You can make the word list in all capital letters so that you can quickly search for an occurrence of the word. If a user needs a definition you can present a data field (using the Form icon in database section) and get a string which can be searched for in the text file and the definition of the word displayed.

Click on the *Object Editor* gadget to enter the Object Editor. Use the Add-Text Window option in the Objects menu to create the window which will display the text. Here you can specify the size, location, the background and border colors of the window and the text, color for highlighting, font, and font size.

You can also define special purpose hit boxes, if you want to permit an end user to scroll text through the window. In addition, the end user needs a hit box for quitting the text window. You have the option of using four text-scrolling commands: *Lineup, Linedown, Pageup* and *Pagedown.* In defining hit boxes that are associated with the text window you use as response strings "lineup," "linedown," "pageup," and "pagedown." You need a hit box with "quit" as the response string to terminate the text window. These response strings are not case sensitive.

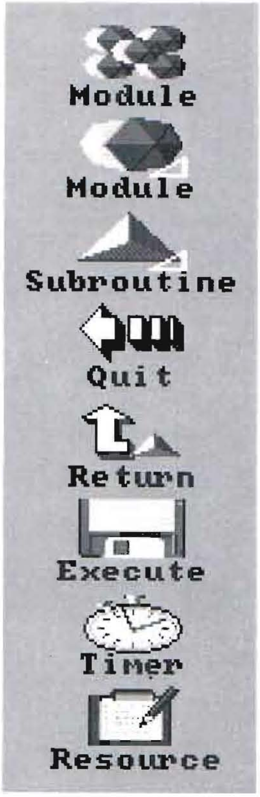
The creation of text windows is described in detail in Chapter 4. If you want to highlight specific phrases you can use |H characters before and after the phrase in your text file. Also if you do not want to reformat columns or tables before presenting through the text window include in your text file the characters |FN before and after that part of the text |FD, the default formatting style, uses each carriage return as a paragraph break for formatting. If you type your text with a text editor, where you hit the carriage return after every line, use two carriage returns for a paragraph break and |FW for forced wrap formatting. You can also create italic, bold and underlined phrases using |I, |B, and |U characters in your ASCII file respectively.

Text windows can be controlled by end users only with mouse hit boxes. Keyboard based controls are not available.

Click in the *Timeout* gadget to specify the amount of time in seconds the text file will be displayed on the screen.

Note: If the Timeout is set to zero, the text file displays for an infinite amount of time. The Timeout can be used as an alternative to a hit box for exiting the text window.

Module Icons



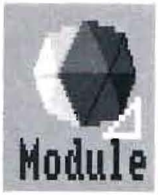
Purpose

The submenu of Module icons give you access to various features that help you manage and control your presentation, including: an icon for exiting the project, one for executing an external program, the subroutine and return icons, and the timer and resource icons.

Usage

See the individual icon descriptions for usage information.

Module Icon



Purpose

The Module icon is used to help organize your application into sections. Module icons can be used as parents for any other icon or group of icons including even other Module icons. You can use Module icons to organize your application into several self-contained modules or sections.

Relation

The Module icon can contain itself and other icons as children. It can also be a child to other parent icons.

Usage

A Module icon is automatically placed at the top of every new Flow Window. It provides a convenient method of organizing your application flow. It is a good practice to organize your application in outline form using several Module icons. Each Module could represent an independent section of the application, such as a unit, topic or lesson. This helps in the design by using the "top down" approach. The "top down" approach implies that you design your general outline first and then fill in the details.

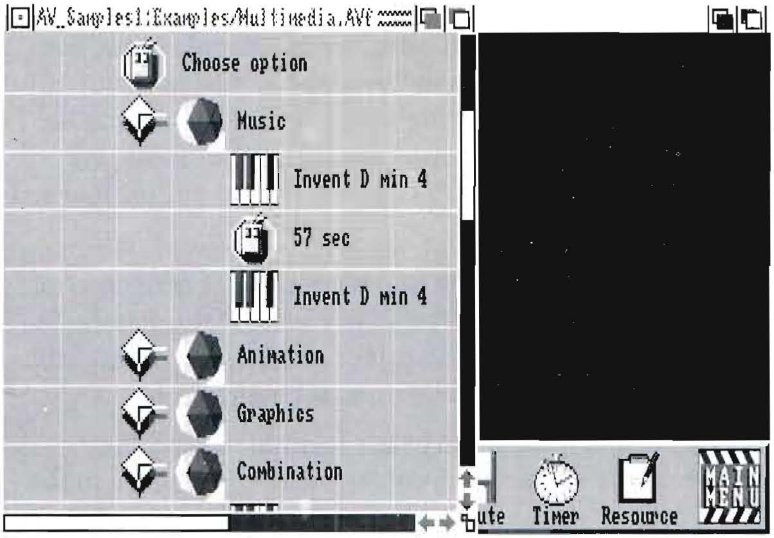
In AmigaVision the Module icon represents the major sections of the outline and its children icons represent the details. You may use the Module icon as a parent and may specify any other icons including other Module icons as children.

You may define variables to be used within the specific module. These will be local variables active for the duration of the module only. Local variables are used only by the children of the module which incorporates them. Outside of the module the local variables defined in the module do not exist.

For example, if you define a variable called LASTNAME in your Module icon and store the string "Smith" in the variable, then it will retain this string until all the actions of the Module are completed. Once the Module is finished executing, AmigaVision forgets the variable called LASTNAME and you are free to reuse the same variable name for any other purpose. Global variables, in contrast, are accessible to all modules of the project which are below them in the outline. Global variables are defined by Variables icons. Global variables can be modified through a Module icon.

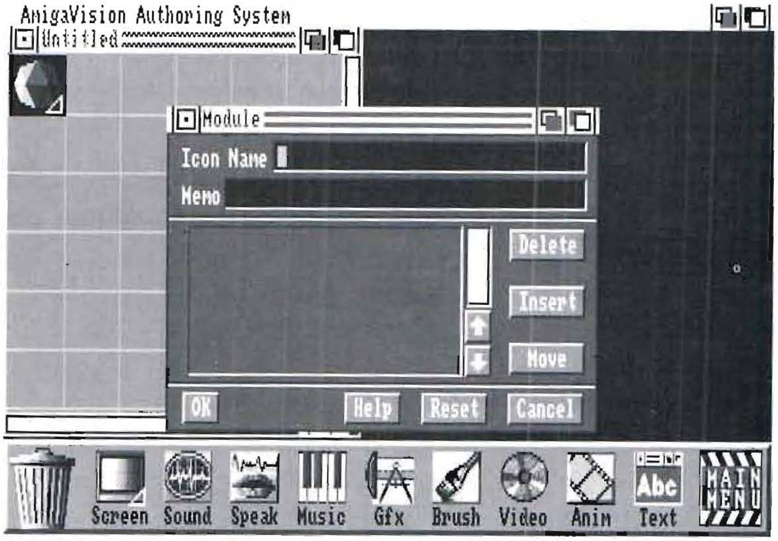
Module icons are also useful for grouping existing icons in the flow window. When you collect a group of icons using the *Collect* option from the **Edit** menu, AmigaVision automatically creates a Module icon as a parent of the group.

Module Example



This example shows the module icons that may contain a single event, a subject, or an entire course. Module icons may contain any other icons as children, including other Module icons.

Module Requester



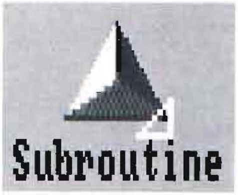
The *Delete, Insert* and *Move* gadgets are used to arrange variable expressions in the list to left of these gadgets. When you click on Insert, the Expression Editor is presented for you to define variable expressions. Here you can either define one or more variables or alter the value of a previously defined variable. When you exit the Expression Editor by clicking on OK, the expressions will appear in the window in the Module requester.

To insert expressions in a particular place in the list of existing expressions, click on the expression below or above the one which you wish to insert. When the expression is highlighted, click on *Insert.*

To delete a variable definition from the list, click on the expression defining the variable displayed in the list. Then click on *Delete.*

To move an expression elsewhere in the list, click on it and then click on the *Move* gadget. To move an expression above its original position, click on the expression above which you want to move. Similarly, to move an expression below its original position, click on the expression below which you want to move.

Double-clicking on an expression presents the variable and the associated expression in the Expression Editor for editing purposes.



Subroutine Icon

Purpose

Use the Subroutine icon to separate and structure a set of actions that you want to use repeatedly in an application.

Relation

The Subroutine icon can contain other icons as children, but cannot contain itself as a child. It must always appear in the left-most column of the flow, i.e., as a sibling of the very first module.

Usage

You can use the Subroutine icon to create sections for repeated use in an application. You can create the section once as a subroutine, and then reference it from various points in the application. You use the Call icon to reference the Subroutine.

It is generally a good practice to use the Exit icon to separate the main modules and the subroutines so that they will be executed only when called explicitly using a Call icon.

If you place a subroutine in the main part of the outline as a sibling to the very first module without an intervening Exit icon, then it will be treated as just another module and its actions will be performed after the completion of the first module.

Just as with the Module icon you may define new variables to be used locally within the subroutine. You may also modify global variables inside the Subroutine icon or by explicitly using the Variables icon.

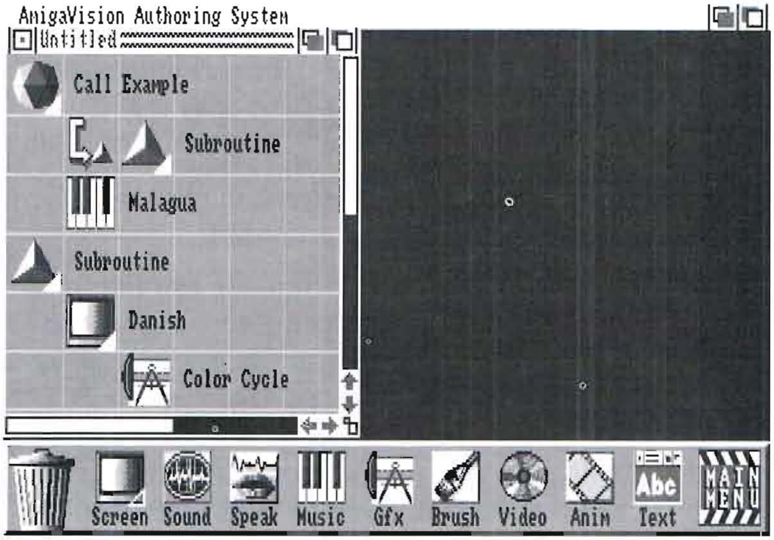
When the Subroutine icon is encountered, the actions of children of the Subroutine icon are performed.

You may also explicitly include a Return icon as a child of a Subroutine icon. When the Return icon is encountered, the application returns to the icon following the original Call icon. For example, you may place a Return icon as a partner to a conditional icon in the Subroutine. If the condition is *true*, then the subroutine is aborted. If the condition is *false*, the subroutine will continue and only after all the subroutine's children have finished executing, will the application return to the icon following the original Call.

When the actions of the Subroutine's last child icon has been performed, the application flow will automatically return to the next icon following the Call icon which referenced the subroutine.

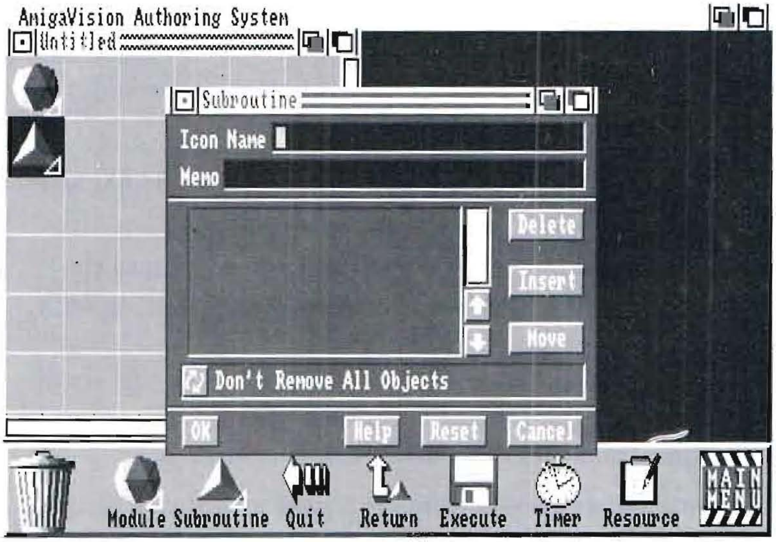
You may place Subroutine icons only in the left-most column of the Flow window.

Subroutine Icon Example



The Subroutine icon can contain any child except another subroutine. It may be either an entire course or only a small segment of a course. You may place Subroutine icons only in the left-most column of the Flow window.

Subroutine Requester



The *Delete, Insert* and *Move* gadgets in the requester are used to arrange expressions in the expressions window to the left of them. When you click on Insert, the Expression Editor is presented and you may define variables. When you exit the Expression Editor by clicking on OK, the expressions will appear in the expressions window.

To insert expressions in a particular place in the list of existing expressions in the window, click on the expression below which you wish to insert. When the expression is highlighted, click on the Insert gadget.

To delete an expression from the list, click on the expression. Then click on the Delete gadget.

To move an expression elsewhere in the list, click on the expression. Then click on the Move gadget. To move an expression above its original position, click on the expression above which you want to move. To move an expression below its original position, click on the expression below which you want to move.

Double-clicking on an expression presents it in the Expression Editor for editing.

Frequently, you may call a subroutine from a point in the application flow which has a screen background along with display objects that were placed on the background by various icons. If you plan to use a new background and a new set of objects within the subroutine, be sure to return the screen to its original state after the subroutine is completed.

For instance, you could store the background image filename in a variable and use it inside a subroutine for returning the screen to its state at which the subroutine was called.

If display objects are on the screen when the subroutine was called, you have the option to either not remove or temporarily remove all of them. Temporarily removing objects means that the objects will be removed only while the actions of children of the Subroutine are executing.

Quit Icon

Purpose

You can use the Quit icon to exit an application and return either to the Amiga Workbench (during runtime) or to the flow editor when creating an application (during preview).

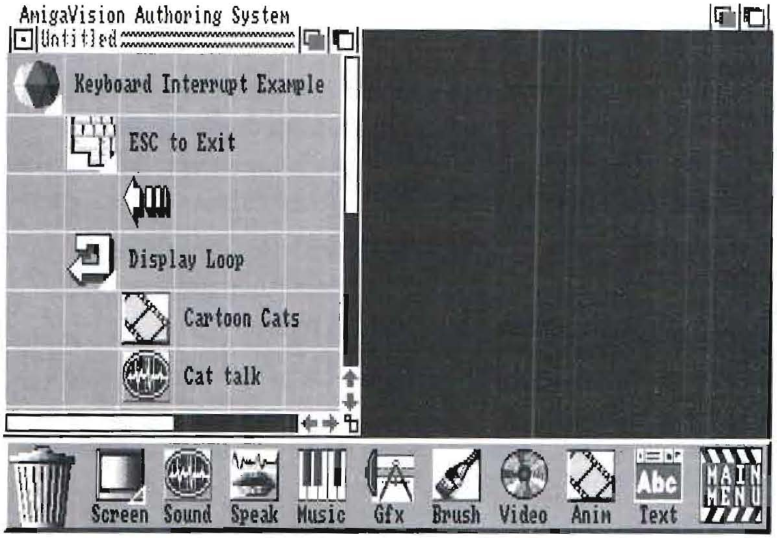
Relation

The Quit icon cannot contain children.

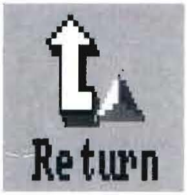
Usage

The Quit icon ends the presentation or course. It returns the user to either the AmigaVision Editor or Workbench screen depending on how the application was executed. If it was executed from the flow editor, it will return control to the flow editor. Alternatively, if the application was executed from a runtime module for execution, then control will be returned to Workbench.

Quit Icon Example



When Esc is pressed the Quit icon ends the presentation of the application.



Return Icon

Purpose

The Return icon explicitly stops the execution of a subroutine and return control back to the next icon following a Call icon.

Relation

The Return icon cannot contain children.

It can appear only as a part of a subroutine. It has no effect if placed in the main flow of an application.

Usage

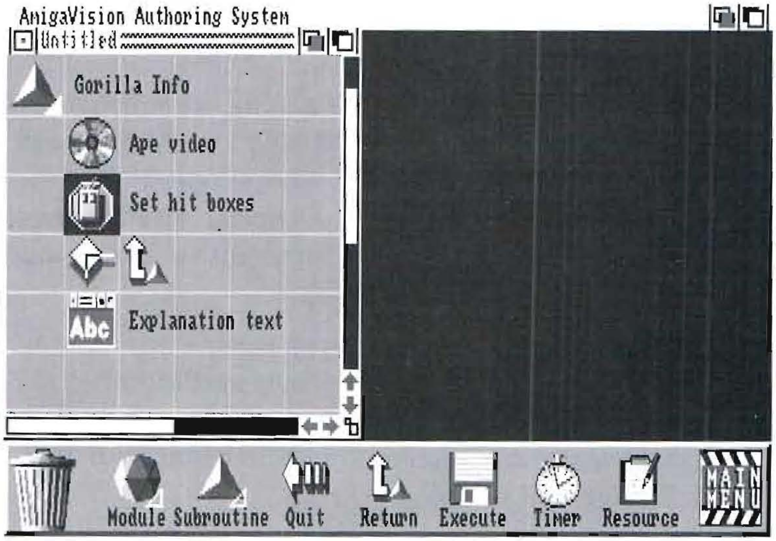
The Return icon is used in conjunction with the Subroutine icon. When the Return icon is encountered, the application exits the subroutine and continues with the icon following the Call icon in the outline which called the subroutine.

If no Return is placed as a child to the Subroutine, the Subroutine will execute completely and then automatically return to the icon following the Call icon.

As one example, you may want to place the Return inside a Subroutine as a partner to a conditional icon. If the condition is true, the Return will be executed. If the condition is false, the Subroutine will continue executing.

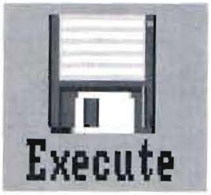
If a Return icon is encountered and there has been no Call, the application will exit as if it encountered a Quit icon. You have to be careful in the design of your application to be sure that a subroutine is not executed except by a Call icon. For instance, if you have a main module followed by a subroutine icon without a Quit icon in between the two, the subroutine will be executed as if it were another module and the application will exit if it encounters a Return icon.

Return Icon Example



The Return is placed as a partner to the If-Then-Else icon. If the condition is true, the application will return to the icon following the Call. If the condition is not true, the Subroutine will continue executing.

Execute Icon



Purpose

The Execute icon references an external program (such as a user-written program, or paint, animation, or sound software) and allows the external program to execute as a part of the application flow. You can also load and execute an ARexx script.

Relation

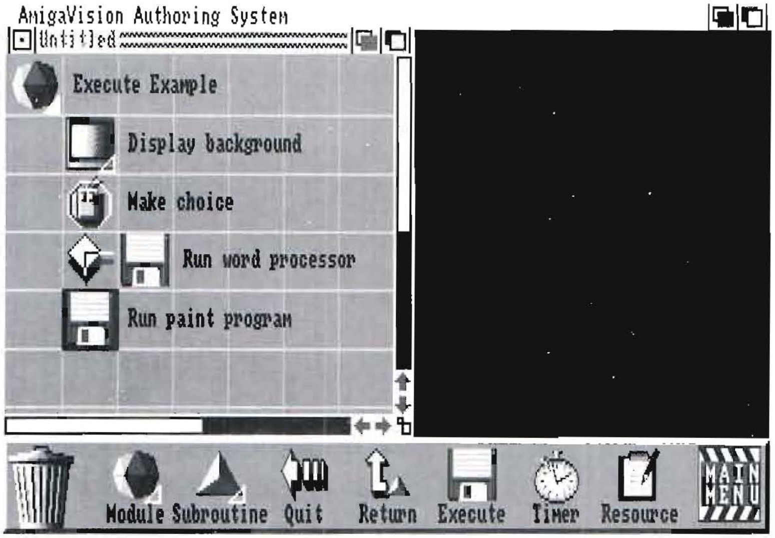
The Execute icon cannot contain children.

Usage

The Execute icon references an external program which runs in Workbench, ARexx or CLI mode. ARexx allows AmigaVision to converse and interact with other ARexx-compatible programs. AmigaVision can launch ARexx scripts and process commands sent to its ARexx port. For details on ARexx, see Appendix B.

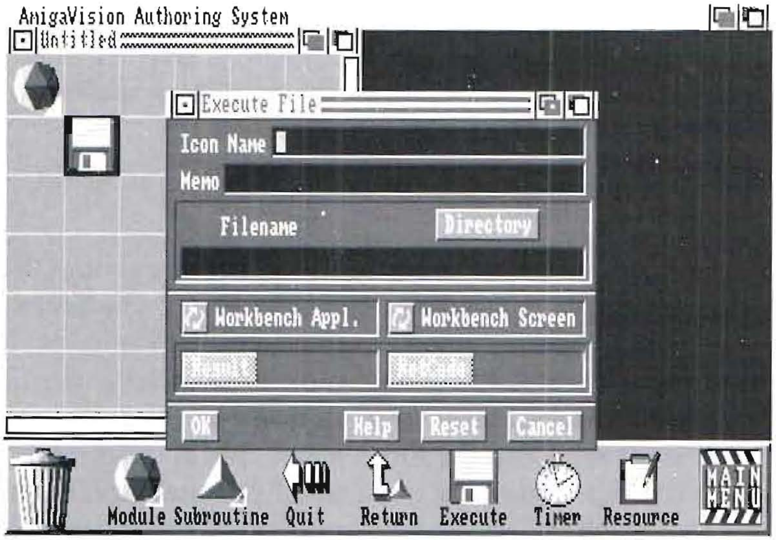
When the Execute icon is encountered, the specified external program is executed and the AmigaVision application is paused. When the external program is exited in its normal way, the AmigaVision application continues executing with the icon following the Execute.

Execute Icon Example



The Execute icon loads an external program and runs it based on the user's choice. After the external program is completed, the application continues with the next icon.

Execute Requester



The *Directory* gadget opens the File requester where you specify the filename of the external program. You may also click on the *Filename* field and type in the path and name of the file to be referenced.

You must specify the screen to be displayed while the external program is executing. The choices are *Custom* and *Workbench.* You select the Custom option if the external program uses a custom screen. If not, you should choose the Workbench option. These two options help AmigaVision manage the display screens properly. For instance, if your external program runs on the Workbench screen, then AmigaVision will display the Workbench screen and move the AmigaVision application screen to the background so that it is hidden from the viewer. Also, if the external program does not need any screen, set to custom screen. This will keep the AV screen visible.

You must also specify the mode of program execution. The Mode options are *Workbench*, *ARexx* and *CLI*.

The *Workbench* option assumes that the external program will be loaded into memory from the Workbench by simulating the double-click of the left mouse button on the Workbench icon. Sometimes a program has options that can be set within the Tooltypes field of its icon info requester (to set graphics mode, default settings, etc.). Using the Workbench option allows you to take advantage of this.

The *ARexx* mode indicates that you are executing an ARexx script.

The ARexx mode allows you to select variables for saving the Return Code and Result values returned by ARexx. The Return Code (RetVal) indicates how successful ARexx was at executing your script. Please refer to the ARexx User's Reference Manual for the meaning of the Return Codes. The Result is the value that the ARexx script itself is returning to AmigaVision. An important note: the ARexx interpreter must be active before you can send any scripts to it. Please refer to Appendix B for more information.

The *CLI* option executes the commands in the string gadget as if they were typed directly into a CLI window. This means you can also include command line settings in the string, for example:

If you had a file called "Text.ex," you could delete it by typing "delete work:amigavision/text.ex" in the file name field.

Timer Icon

Purpose

You use the Timer icon to time specific parts of an application. It does not stop the application, but merely acts as a stopwatch. The Timer measures time in elapsed seconds with up to two decimal places (e.g., 100.34 seconds).

Relation

The Timer icon cannot contain children.

Usage

You may have up to nine timers running simultaneously. They are identified by numbers from 1 to 9. You may start, stop and restart the timers as necessary.

The Timer is like a stopwatch. When started, it begins counting elapsed time. When you stop the timer, it retains the elapsed time. If you restart the timer again, it continues counting from where it left off. If you want it to start counting from zero again then you must choose the Start option instead of Restart.

Each of the nine possible timers is identified by a number. You can read the value of a specific timer into a variable by using the timer() function in the Expression Editor. This function returns the elapsed time of a specific timer number in seconds.

For example, you might create a variable like TestTime = timer(2). This would store the current value of timer #2 to the variable called TestTime.

You can also directly use the timer() function in a test condition such as in an If-Then-Else icon:

```
timer(2) >= 10
```

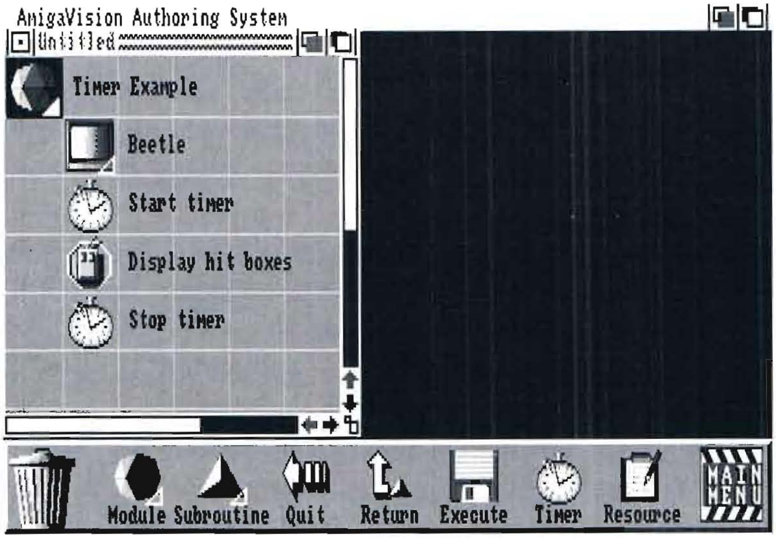
This expression is interpreted as: if timer number two indicates an elapsed time that is greater than or equal to 10 seconds, the If (True) part of the condition is performed and the Else part is ignored.

In general it is better to use >= or <= instead of == when testing the timer condition, as the timer is not updated at every second and it may miss the value while sampling around it. For instance if you state a conditional expression:

timer(1) == 10.00

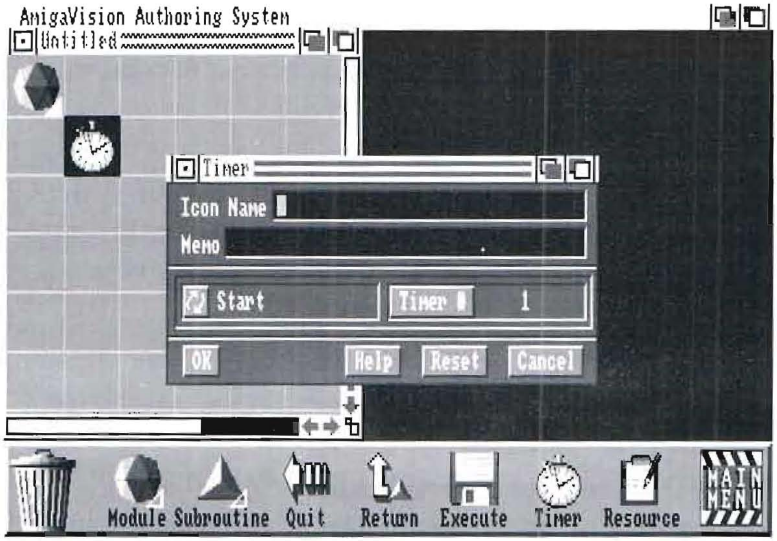
then the condition might never be True since the timer clock may jump from 9.95 seconds to 10.01 seconds and miss the 10.00 second mark entirely. Thus it is prudent to use the >= or <= tests when you state conditions using the timers.

Timer Icon Example



In this example the timer can be used to time a student while answering questions or reading text. You can start, stop or restart it. You may have all nine timers operating simultaneously.

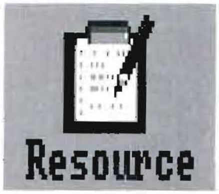
Timer Requester



Icon Name
Memo
Start
Timer # 1

Click on the *Start/Stop/Restart* gadget to select the appropriate action. Click on the Timer # field to enter the number of the timer being used. The timers are numbered from 1 to 9. This opens the Specify Value requester.

Resource Control Icon



Purpose

You can use the Resource Control icon to pre-load and unload resources such as picture, sound, animation and music into memory. This reduces long waits in the middle of a presentation.

Relation

The Resource icon cannot have any children.

Usage

The Resource Control icon allows you to pre-load into memory, image files, sound files, and other files used in the presentation. This will make the application respond rapidly during execution of audio-visual events.

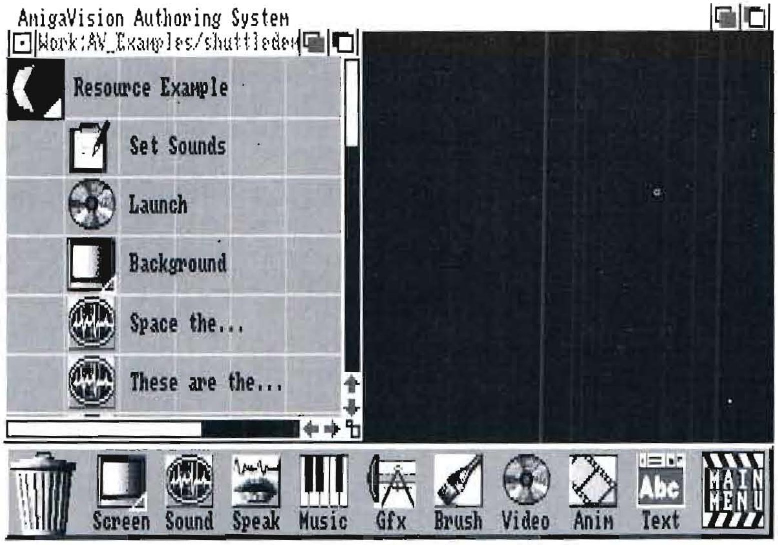
When you load resources using this icon, there will be fewer disk accesses and a consequent decrease in the time required for presentation. For this reason, loading with the Resource Control icon is also useful when a certain picture or music file is used repeatedly in the application.

Resources which are loaded with this icon will stay in memory until they are unloaded by another Resource Control icon or until the application ends.

Note that you must use one Resource Control icon for loading and another for unloading the same resources. If you fail to unload resources after use then you may run short on memory and may not be able to load other resources. AmigaVision discards unused files if memory gets low. It is a good practice to specify explicitly the load and unload operation, as it would free up memory for new resources to be loaded.

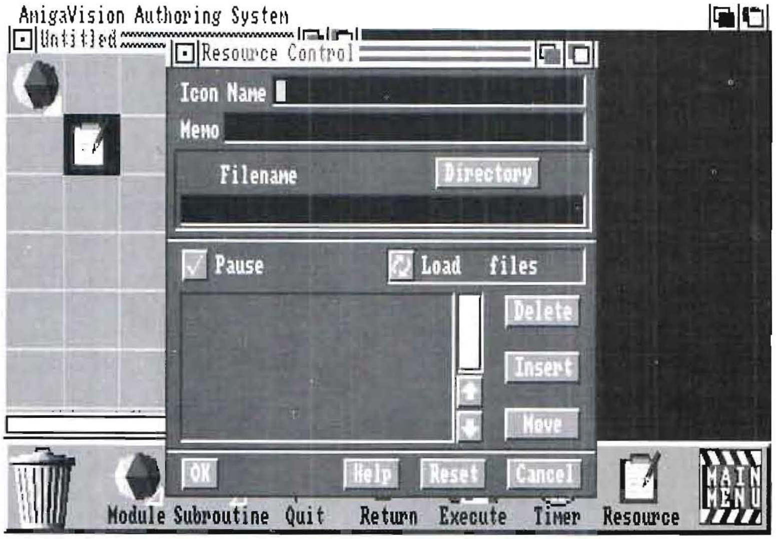
During execution, AmigaVision first checks for the presence of resource files in memory. Files which cannot be found are loaded from the specified file on your disk.

Resource Control Icon Example



In this example the Resource Control icon is used to load the files for the Sound icons. The sound file starts almost immediately the second time it is instructed to play, because it is already in memory.

Resource Control Requester



When you click on the *Directory* gadget, the File requester is presented. Here you may specify the volume, drawer and name of the resource files. If you want this file to be entered into the resource file list, then click on the Insert gadget to the right of the window.

To insert a resource in a particular place in the list of existing resources in the window, click on the resource below which you wish to insert. Then, click on *Insert* to place the resource from the Directory field into the Resource Window.

Also you can edit a specific filename in the list. Double-click on the resource. Its name will be put in the Filename field for you to edit. You can then insert the edited name back in the list.

To delete a resource from the list, click on the resource. Then click on *Delete.*

To move a resource elsewhere in the list, click on the resource. Click on *Move.* To move a resource above its original position, click on the resource above which you want to move. To move a resource below its original position, click on the resource below which you want to move.

Select the *Pause* button to cause the application to wait until the files are loaded before executing the next icon.

Select the action by clicking on the Load or Unload gadget. Usually when you need to unload previously loaded files, you can simply use the Copy feature from the pull-down menu and copy the Resource icon that was used to load. Selectively delete the files that you do not want unloaded and then select the Unload option. This saves you the labor of specifying each of the files to be unloaded.

Trashcan Icon

Purpose

The Trashcan icon is used to discard icons from the outline

Relation

The Trashcan is only a menu icon. It cannot be placed in the outline.

Usage

The Trashcan icon is always present at the left side of the icon menu. To discard an icon from the outline, drag it to the Trashcan and drop the icon.

Caution: When you discard a parent icon which contains children, the children are discarded as well. A warning message appears to allow you to change your mind before discarding a parent icon. As there is no "undo," be careful discarding parents with children.

Return to Main Icon

Purpose

The Return to Main icon is used to return to the main icon menu. It is only a menu icon and cannot be placed in the outline.

Relation

Return to Main is a only a menu icon. This cannot be placed in the outline.

Usage

Return to Main icon is present on the right side of all the icon submenus. To return to the main menu of icons, click on Return to Main. The icons in the main menu are Control, User Interrupt, Database, Wait, Audio Visual, and Module.

Appendices

Appendix A: Error Messages

Listed below are typical AmigaVision error messages.

Amiga is low on memory; please free some and try again.

AmigaVision needs more RAM to perform the operation. Close down the Workbench or other programs you are running, or any open windows that aren't needed at the time.

Call Icons may only reference Subroutine Icons:
Continue: Select another icon
Cancel: Terminate referencing process.

You tried to reference something other than a subroutine from a Call Icon. Use Goto if you wish to jump to a section of the flow which is not a subroutine, or select a subroutine to reference.

Cannot attach icon at this position; place on the right side for partner; to the right and down one for child; or on top of sibling you wish to insert before.

Placement of icons in a Flow or Content Window is fairly precise. See Icon Relations for more information on placing icons.

Cannot edit a referenced icon; edit actual icon.

If you wish to define an Icon Action, you must select the actual icon, not a Goto or Call partner icon.

Cannot create — Filename is not specified.

You must specify a filename for the database before clicking the Create gadget.

Cannot initialize video player. Please check player, cable, and settings.

Check the physical connection of the videodisc player to the computer as well as the device and baud rate settings in Video Setup.

Cannot insert — A Boolean has not been set.

Before a record can be inserted in the database, all Boolean fields must be set.

Cannot insert — Database already contains record with this key.

Once a field is defined as a key field, its name cannot be defined as a field a second time.

Cannot insert — Empty keys are not allowed.

At least one field that is defined as a key field must be set before a record may be inserted.

Cannot insert — Record length would exceed maximum size of 4000 characters.

The new field cannot be added to the database because it would make the combination of all the fields exceed 4000 characters. Try a smaller field or shrink one of the others.

Cannot insert as a key — Max. Key length of 100 would be exceeded.

The field you chose to be a key has a length greater than the 100 character maximum.

Cannot open/create — More than 10 Databases open.

When presenting a RunTime file this error will occur if the file tries to open more than ten databases at the same time.

Cannot open screen.

Not enough memory to open the desired screen. Try freeing some memory or use a lower resolution image.

Cannot open window.

Not enough memory to open the desired window. Try freeing some memory by closing other programs or windows.

Cannot rubberband from this point.

This notice will appear if you attempt to collect in an area that contains no icons.

Clear the background image?

Asks for confirmation to the Pull-Down Menu in the Object Editor "Project-Screen Clear."

Configuration process aborted: File(s) have not been copied/ created.

When copying videodisc drivers from floppy, this message will appear if you did not specify a valid path to the drivers. These drivers are normally stored on the AmigaVision_Boot disk in the DEVS:PLAYERS directory. It can also occur when attempting to install the videodisc driver "player.device" onto the hard disk from floppy if you specify the wrong path, which is normally DEVS: on the AmigaVision_Boot disk. This error can also occur if there is not enough space on your hard disk to copy the driver to.

Content Window can only contain Audio Visual icons; requested placement ignored.

Only audio visual icons can be placed in a Content Window.

Date is invalid.

You need to enter dates in the format Month/Day/Year; e.g., "3/15/66" or "03/15/1966."

Delete all display objects?

Asks for confirmation to the Pull Down Menu in the object editor "Project-Clear All."

Destination filename needed.

When using Install you must specify a Destination filename before clicking OK.

Devs:Players subdirectory needs to be created.

When installing the videodisc driver, AmigaVision must create a drawer, "Players," in the DEVS: directory to store the videodisc drivers.

Divide by zero. Please check variable values.

You cannot divide a number by zero.

E Form icons can only be placed as children of Form icons.

The Exit Form icon is only applicable when used with a Form icon.

Existing icon is already linked to another.

You cannot replace a reference icon by dropping a new icon on top of the old one; first throw away the old reference icon.

Field name is empty or contains illegal characters.

Fieldnames can only contain alphabetic characters, numerals and the underscore "_" character.

File is edit-protected; cannot edit.

After creating a RunTime file with the edit protection option turned on, you will not be able to reload the RunTime file for editing.

File or Drawer is too long.

The name of the path or file is too long. A path cannot exceed 255 characters and a filename cannot exceed 32 characters.

Goto icons terminate execution of sibling list: icons below this point will be ignored.

Icons placed below a Goto icon will not be executed since the Goto icon jumps to a separate part of the flow and does not return.

Icon may not reference itself:
Continue: Select another icon
Cancel: Terminate referencing process.

A Goto may not reference itself.

Icons which are children of 'AV' or 'Wait' icons may not be collected.

When you collect icons, those icons are placed as the children of a new module. Because 'AV' and 'Wait' icons may not be parents of 'Module' icons, you may not collect the children of 'AV' or 'Wait' icons.

Image output only possible on printer.

To print a graphic version of your flow you must have a printer.

Incompatible display objects ignored.

If you enter the Object Editor from the Editors menu, you may not create text windows or text fields since these may only be defined from the Text and Field icons.

Incorrect number of parameters.

When inserting a formula, this error will occur if you do not supply every parameter needed in the equation.

Incorrect parameter type.

You may not put strings in expressions that require a numeric value.

Insufficient memory.

Not enough memory for the selected operation. Try to free some memory or choose a different operation.

Invalid key-string or missing separator detected at marked position.

When defining a keyboard interrupt or wait, this error will occur if you do not separate multiple keys with commas.

Missing parentheses.

Formulas must appear exactly as they do in the list in regards to their format, including the parentheses.

(# of deleted references) Illegal references were corrected.

If an icon that has been referenced or deleted, or if referencing icons have been moved or copied and the reference is no longer legal as seen by the evaluator, the references will be deleted counted and this message will be presented along with the count.

No Fields specified for Database.

You must define at least one field before creating a database.

Number will not fit in field when properly formatted.

This error will occur with floating point numbers that, when displayed, will be larger than the field space allotted for them.

Numeric operation on a string or a Boolean variable is not allowed.

Expressions that require numeric value will not function on string or Boolean data.

Once a Subroutine icon is placed, all other types of icons in left-most column and below are ignored.

After starting to define subroutines, only other subroutines may appear in the left-most column of the flow.

Only 'AV' icons may be attached at this position.

You can only place Audio Visual icons as children of Screen icons.

Only the following icons can be referenced: Icon's siblings; Icon's parent and grandparents; Siblings of icon's parent; Siblings of icon's grandparents.
Continue: Select another icon.
Cancel: Terminate referencing process.

The Goto and Conditional Goto icons may only reference icons to their left on the parent module flow as well as other icons that appear in the left-most position of the Flow Window.

Only 'Wait' icons may be attached at this position.

You can only place Wait icons as children of a Grouped Wait icon.

Please specify a RunTime filename.

You clicked OK without first specifying a RunTime filename.

Presentation error: Could not open screen; aborting.

During a presentation AmigaVision did not have enough memory to open a new screen.

Referencing process cancelled.

This message appears whenever there has been a mistake in referencing and you choose "Cancel."

Referencing process continuing.

This message appears whenever there has been a mistake in referencing and you choose "Continue."

Return icons may only be placed as descendants of Subroutines.

The Return icon is only valid as part of a Subroutine module.

Screen resolution and interlace settings of the consecutive pictures must be the same to perform specified transition. (Fade transitions will work)

You cannot do this particular picture transition unless the two pictures are in the same mode. Screen modes are a combination of the resolution (low-res, high-res, extra halfbrite, Hold & Modify) and Interlace (on or off). Both screens must match both parameters for this transition to work.

Source file name needed.

When using RunTime Install you must specify a Source filename before clicking OK.

Specified file is not an AmigaVision application.

This error occurs when attempting to load a file that is not in AmigaVision format.

Specified file is not an AmigaVision 'Display Object' file.

The file chosen was not a collection of display objects saved from the AmigaVision Object Editor.

Specified file is not an ILBM or ANIM5 format.

This error occurs when attempting to load a picture that is not in the IFF format that AmigaVision supports.

Specified file is not ASCII.

This error occurs when defining a text file that is not an unformatted, ASCII file.

Specified file is not of ANIM5 format.

This error occurs when attempting to load an animation that is not in the IFF format that AmigaVision supports.

Specified file is not of 8SVX format.

This error occurs when defining a digitized sound that is not in 8SVX (one-shot) or IFF instrument format.

Specified file is not of IFF format.

This error occurs when attempting to load a picture or animation or sound file that is not in IFF format.

Specified file is not of ILBM format.

This error occurs when attempting to load a picture that is not in the IFF format that AmigaVision supports.

Specified file is not of SMUS format.

This error occurs when attempting to load music that is not in the IFF format that AmigaVision supports.

Subroutine icons must be placed in the left-most column.

Subroutines may only be defined in the left-most position in the Flow Window.

Syntax error. Please refer to the User's Manual.

This message will appear when you enter an expression that cannot be evaluated. Please refer to Chapter 4 for a description of the Expression Editor.

There are no logical operators in this conditional expression.

If the Expression Editor is entered on a condition, you must use logical operators.

There is no expression. To exit — click on CANCEL.

This requester appears when you click on the OK gadget without having defined any variables or expressions.

Type Mismatch of variables in expression.

Certain functions and assignments depend on particular variable types. You cannot assign string data to a numeric value. You cannot perform the absolute value function on a string. Refer to the function list to see which expression require which data type.

Unknown variable. Please define variable before using.

You attempted to use a variable that hasn't been declared in the expression. Choose a new variable or create the variable you wish to use.

Value cannot be negative.

The number specified in this field must be greater than or equal to zero.

Value is invalid.

The number you have entered is too large or too small or contains non-numeric characters.

Value is not the correct type.

Strings may not be placed in numeric fields.

Value is out of range.

The number you entered is less than or greater than the limits for this field.

Variable name too long.

Variable names cannot exceed 20 characters.

Video Error: Cannot open Devs:Player.device. Please install proper driver software.

You have not used the Project Pull Down Menu "Video Setup" which will install the "player.device" file in the DEVS: directory. This is necessary before performing any videodisc actions.

Warning: Deleting this icon will also delete its children.

When you attempt to throw away a parent icon, this warning will allow you to abort throwing away the parent if it has children.

Warning: Deleting this icon will also delete its partner.

When you attempt to throw away a referencing icon, this warning will allow you to abort throwing away the referencing icon and its partner.

Warning: Insufficient or Fragmented memory! Free Memory and hit Continue to try again, or Cancel to Abort AmigaVision.

When you present your flow, AmigaVision closes down its editor to free up extra memory. If memory becomes fragmented or too much is used, AmigaVision will be unable to re-open its editor. Close other applications and/or windows to free memory or else AmigaVision will exit to the operating system.

You cannot match two left-partner icons; place a Module icon as the right, then add other icons as its children.

If-Then, Goto, Conditional Goto and Call are considered "left-partner" icons because they are mated with the icon to their right. With an If-Then icon you cannot place a left-partner icon as its mate. If you wish to include such icons you must place a module icon and place the left-partner icons as children of that module.

You must choose at least ONE destination diskette!

You must select at least one floppy drive from the "Copy resources to:" list to create the RunTime version.

You must enter a Course Name!

You clicked OK without first specifying a Course Name.

You must enter a Filename!

This message will appear if you do not specify a filename in the "Create RunTime" requester.

You must specify a file or Cancel.

In the Specify Filename requester, this warning will appear if you click OK and the Filename textbox is empty.

Appendix B

ARexx Interface

ARexx is the Amiga version of REXX, a powerful yet easy-to-learn general programming language. ARexx is available from Commodore Amiga as part of the Workbench 2.0 System Software.

This chapter will assume you are familiar with ARexx as documented in the manuals that came with your Amiga. For more complete coverage of the REXX language in general, your best choice is *The REXX Language: A Practical Approach to Programming* by M. F. Cowlishaw, the inventor of the language.

A unique feature of the ARexx language (and REXX in general) is the ability to communicate with other programs, often referred to as Inter-Process Communication, or IPC for short. To make use of IPC, a program must be ARexx-compatible. An ARexx compatible program can exchange information with any other compatible program, using ARexx as a hub. AmigaVision is fully ARexx-compatible: it can both send and receive commands using ARexx IPC.

The advantage of IPC is that you are not limited to the capabilities of a single program. With ARexx acting as a hub, you can connect many powerful special-purpose programs together, resulting in an integrated super-program of enormous power. For example, AmigaVision could control a complex number-crunching program that could send its results to an image-generating program that could send its images back for AmigaVision for display. The user need never know that several different programs were involved.

Sending Commands To ARexx

Commands are sent to ARexx using the AmigaVision **Execute** icon. Refer to that section for more information on how to use the icon. ARexx programs that are started by other programs like AmigaVision are often call "scripts" or "macros."

For AmigaVision to be able to interact with ARexx, the support library rexxsyslib.library must exist in your LIBS: directory when AmigaVision is started.

The ARexx interpreter must be running before you can send any commands to it. Many people start ARexx during their Startup-Sequence, so it is ready after every reboot. To do this place "rexxmast" as the last line of your startup-sequence. You could also start ARexx from inside AmigaVision by using an Execute icon, in CLI mode, to send the command "rexxmast."

The Execute icon, when in the ARexx mode, sends the string in the *Filename* gadget to ARexx for processing. That string is usually the name of an ARexx script file to be executed. If you do not specify a file extension, ARexx will assume the default AmigaVision extension of .AV. For instance, the script filenames MyScript and MyScript.AV are equivalent when sent to ARexx from AmigaVision.

The **Execute** icon's Filename string can also be used to send what ARexx calls a string file. In this case, the string consists of actual ARexx instructions comprising a complete, tiny ARexx script, and not the name of a file. To indicate to ARexx that this is a string file, you enclose the string in quotes. For example, the string "do for 5; say 'HELLO'; end" will cause ARexx to say "HELLO" five times.

When ARexx scripts are launched using the **Execute** icon, the default host address is set to the AmigaVision port, AV.REXX.

After ARexx has finished its processing, it may have a Return Code and a Result to return. That information will be stored in the AmigaVision variables you specify in the Execute icon's RetCode and Result fields. The Return Code is usually a numeric value from ARexx indicating how successful it was in executing your command. The Result is a value returned by your script using the RETURN or EXIT instruction. (Note: The AmigaVision/ARexx commands also provide a method for passing values back to AmigaVision, making the Result variable field in the Execute icon somewhat redundant. You may choose whichever best meets your needs.)

If AmigaVision cannot communicate with ARexx while processing an Execute icon, it will set the RetCode variable to a string describing the problem. For example, if AmigaVision did not find rexxsyslib.library during its startup, the RetCode variable will be set to NO REXXSYSLIB. If ARexx is not active, the RetCode variable will be set to NO REXX.

Receiving Commands From ARexx

Since the AmigaVision commands deal primarily with variables, and variables only have meaning during runtime, AmigaVision will only process commands during runtime. Commands received at any other time will be returned with an error return code.

```
    'GETVAR CUSTNAME' /*AmigaVision Command*/
    NAME = RESULT /*ARexx command*/
```

Before a program or script can send commands to AmigaVision, it must set up ARexx to perform the communication. First, it must tell ARexx where to send the command. This is called the host address. The following instruction sets the host address to AV.REXX, the AmigaVision ARexx port:

```
ADDRESS AV.REXX
```

AV.REXX must be in all upper-case letters. If the script sending commands to AmigaVision was launched by an Execute icon, the host address will already be set to AV.REXX (but it never hurts to state it explicitly).

Next, if the program expects ARexx to return results from AmigaVision, it must indicate that with the ARexx instruction:

```
OPTIONS RESULTS
```

Now ARexx is ready to send commands to AmigaVision.

The AmigaVision/ARexx Command Set

This section describes the commands AmigaVision will process when sent to its ARexx port, AV.REXX. After each command is processed, AmigaVision will set the ARexx variable RC to indicate the success of the command. See the next section for descriptions of the return codes.

SETVAR

Usage: SETVAR variable value

The SETVAR command sets an AmigaVision variable to the specified value. All values are assigned to AmigaVision variables as strings. Any numeric values assigned using SETVAR will have to be converted using the AmigaVision function Integer() or Float() before they can be used as numbers. Also, you should be familiar with the way ARexx captializes values it processes, or you may get unexpected results.

Examples:

SETVAR NAME1 Joe Smith will assign the string "JOE SMITH" to the AmigaVision variable NAME1. ARexx evaluates the variables "Joe" and "Smith" and assigns the concatenated value to NAME1. If "Joe" and "Smith" have not been assigned values, the result will be "JOE SMITH".

SETVAR NAME2 "Joe Smith" will assign the string "Joe Smith" to the AmigaVision variable NAME2. The quotes preserve capitalization.

SETVAR AMOUNTSTRING 23.45 + 100 will assign the string 123.45 to the AmigaVision variable AMOUNTSTRING. If you need to use the value of AMOUNTSTRING as a number, you must convert it by assigning the numeric value to another variable, then using that new variable. For example, using an XY icon, you could perform the assignment:

AMOUNTNUMBER = Float(AMOUNTSTRING).

GETVAR

Usage: GETVAR variable

The GETVAR command retrieves the value of an AmigaVision variable. The value will be returned in the ARexx variable RESULT.

Remember: To get any results from AmigaVision, you must have previously told ARexx you want them, using the instruction OPTIONS RESULTS. If you don't, this command will appear to do nothing.

Example:

GETVAR LASTNAME will assign the value of the AmigaVision variable LASTNAME to the ARexx variable RESULT. Note: Since many commands store results in RESULT, you should set one of your other variables to RESULT as soon as possible.

```
Example: GETVAR LASTNAME
         Lname = RESULT
```

AVVERSION

Usage: AVVERSION

The AVVERSION command retrieves a string describing the AmigaVision version. The string will be returned in the ARexx variable RESULT.

Remember: To get any results from AmigaVision, you must have previously told ARexx you want them, using the instruction OPTIONS RESULTS. If you don't, this command will appear to do nothing.

AmigaVision/ARexx Error Return Codes

After each command is processed, AmigaVision sets the ARexx return code variable RC to a value indicating the success of the command. If RC is zero, the command executed successfully. The following list describes the meaning of non-zero RC values.

5 Command is recognized, but could not be performed.
Example: GETVAR from a non-existent variable.

10 Command is not recognized.

15 AmigaVision is not processing commands now.
Commands are only processed during run-time.

20 AmigaVision is closing.
Command was received during AmigaVision's shutdown process. This is a warning that the port AV.REXX will disappear shortly!

Examples: AVR and Fun With ARexx

The ARexx script AVR.REXX is handy for experimenting with ARexx and the AmigaVision/ARexx command set. Run it from a CLI using the command

```
rx avr.rexx
```

While in AVR, you have access to all ARexx instructions.

Any ARexx variables you create are preserved until you terminate AVR. AVR will display any results and explain any return codes AmigaVision may return.

The AmigaVision flow FunWithARexx is a companion for AVR. It has two variables, A and B, that you can access using AVR. FunWithARexx also demonstrates how to start and access ARexx from AmigaVision. (FunWithARexx assumes that the ARexx startup command "rexxmast" can be found in the current command path.)

```
/* AVR.REXX (MEW) 900213 */
/* AVR is an interactive program for experimenting with ARexx and */
/* AmigaVision's ARexx commands. At the AVR> prompt, you can enter */
/* any valid ARexx clause, including AmigaVision commands. AVR will */
/* immediately execute those statements and display any results. To */
/* terminate AVR, type EXIT at the AVR> prompt. */
/* These constants let us control the console's text colors  */
csi = ''9b''x                                        /* Control Sequence Introducer */
normal = csi || ''31:40m''
reverse = csi || ''33:41m''
hilite = csi || ''32:40m''
say '' ''
say '' ''
say reverse ''AVR: Interactive AmigaVision/ARexx Console'' normal
say '' ''
say '' (Type EXIT to terminate)''
say '' ''
address AV.REXX                           /* Send commands to the AmigaVision ARexx port. */
options prompt hilite || ''AVR>'' || normal || '' ''                       /* A nice prompt. */
options results                                            /* We want results back from AV. */
options failat 21                        /* AV errors can go up to 20; we'll handle them . . . */
signal on syntax                                 /* . . . using our own ARexx error handler. */
do forever
  parse pull avcommand                                          /* get a command from the user */
  drop rc                                                   /* clear out previous return code */
  drop result                                                    /* clear out previous result */
  interpret avcommand                                                  /* execute the command */
  /* Variable 'result' contains the result of our command. */
  /* We'll only print 'result' if we actually got something. */
  if ( symbol('result') = = 'VAR' ) then say ''The result is:'' result
  /* Variable 'rc' contains the return code from AmigaVision. */
  /* Explain what the codes mean . . . */
  if ( symbol('rc') N= = 'VAR' ) then nop   /* do nothing: rc was not set */
  else if ( rc = 0 ) then nop        /* do nothing: no error occured */
  else if ( rc = 5 ) then say ''*** AV Error 5: That command couldn't be performed''
  else if ( rc = 10 ) then say ''*** AV Error 10: That command wasn't recognized''
  else if ( rc = 15 ) then say ''*** AV Error 15: Commands only processed during run-time''
  else if ( rc = 20 ) then say ''*** AV Error 20: AmigaVision is closing!!!''
  else say ''*** AV Error'' rc '': UNKNOWN''
  say '' ''
end
syntax:/* Our own ARexx handler for fatal errors */
if ( rc = 13 ) then say ''*** AmigaVision port AV REXX not found ***''
else say ''*** Error'' rc ''at line'' sigl ''of AVR.REXX ***'' exit
```

A Sample Application Script

As mentioned earlier, ARexx's Inter-Process Communication (IPC) provides a method for seamlessly integrating programs with special capabilities. The best way to illustrate that is by example.

Let's begin by inventing a hypothetical application: We want to create a point-of-sale kiosk that will allow the public to order merchandise using a credit card. Using AmigaVision, we can create a flow to collect the buyer's name, credit card, and purchase information, and save it in a database file. However, we need to authorize this use of the credit card. With the problem defined, let's make some assumptions:

—We have a specialized ARexx-compatible program that, given a person's name and credit card information, will provide an authorization code. Perhaps this program uses some kind of telephone hookup to access a bank's database. Let's say this program's ARexx port name is CREDIT and it accepts an AUTHORIZE command.

—We have two AmigaVision variables, called CUSTNAME and CARDNUM, that contain the customer's name and credit card number.

—We have an AmigaVision variable, called AUTHNUM, to store the authorization number.

Given this, we can create an ARexx script to connect the CREDIT program to AmigaVision. This script would be sent to ARexx using the AmigaVision Execute icon.

```
/* Authorize.AV (MEW) 900212 */
/* An AmigaVision/ARexx script to obtain credit card authorizations */
/* from a hypothetical program called CREDIT */
/* This script would only be launched by the AmigaVision Execute icon, */
/* so this line is redundant. But just in case . . . */
      ADDRESS AV.REXX
/* This informs ARexx that we want results returned. */
      OPTIONS RESULTS
/* Get the customer name from the AmigaVision variable CUSTNAME */
      'GETVAR CUSTNAME'
/* The name is in RESULT; save it in the ARexx variable NAME */
      NAME = RESULT
/* Get the credit card number from the AmigaVision variable CARDNUM */
      'GETVAR CARDNUM'
/* The card number is in RESULT; save it in the ARexx variable CCNUM */
      CCNUM = RESULT
/* This is the command to our hypothetical authorization program  */
/* Let us assume that its port name is CREDIT and the authorization */
/* number comes back in the ARexx variable RESULT. */
      ADDRESS CREDIT AUTHORIZE NAME CCNUM
/* The authorization number is in RESULT; let's save it in ANUM */
      ANUM = RESULT
/* Now send it back to AmigaVision's variable AUTHNUM  */
/* Notice that ANUM is not in quotes; this means that the value */
/* of ANUM is sent to AmigaVision. */
      'SETVAR AUTHNUM' ANUM
/* That's all, folks! */
```

Appendix C: Keyboard Shortcuts

AmigaVision supports keyboard shortcuts for many of its operations in the Flow Editor and the Object Editor. All of these commands are accessed by holding down the Right-Amiga (*A*) key, which is to the right of the spacebar, and pressing the corresponding key. For instance, to Save a file you hold the *A* key and press S to save your file. You do not need to press the Shift key to capitalize a letter.

Flow Editor Keyboard Shortcuts

Project Menu

***A* N** New Flow Window.

***A* Y** New Content Window.

***A* L** Load—Load an AmigaVision file for editing.

***A* S** Save—Save the current AmigaVision file; overwrites any existing file.

***A* A** Save As—Save the current AmigaVision file after specifying a new name.

***A* X** Defaults—Set the default paths for Audio Visual files.

***A* P** Print—Print the current AmigaVision file to a printer or file.

***A* .** Present . . . —Show the current AmigaVision file as a presentation.

***A*]** RunTime—Create.

***A* [** RunTime—Install.

***A* U** Video Setup—Configures videodisc settings.

***A* Q** Quit—Exit AmigaVision.

Edit Menu

A O Collect—Toggle on/off Collect mode for organizing icons.

A C Copy—Toggle on/off Copy mode for duplicating icons.

A I Info—Display an icon's settings (same as double-clicking an icon).

A = Preview—Preview an icon's actions (same as clicking Preview from an icon's settings requester).

A T Telescope—Contract or expand the children of the selected parent icon.

A , Search—Search for named icon in the window.

Tools

A E Object Editor—Enter the Object Editor.

A V Videodisc—Enter the Videodisc control panel.

A D Database—Enter the Database Editor.

Object Editor Keyboard Shortcuts

Project

A L Load—Load an Object Editor file.

A P Preview—Preview all objects as they will appear in the presentation.

A R Redisplay—Redraw all objects on the screen.

A S Save—Save the current Object Editor file.

A A Save As—Save the current Object Editor file after specifying a new name.

A H Help—Display help for the Object Editor.

A E Exit—Exit to the AmigaVision editor.

Objects

A C Copy—Duplicate an object.

A D Delete—Remove an object.

A M Move—Move an object on the screen.

R Rectangle—Adds a rectangle object.

P Polygon—Adds a polygon object.

L Line—Adds a line object.

C Circle—Adds a circle object.

E Ellipse—Adds an ellipse object.

T Text/Variable—Adds a text/variable object.

B Brush—Adds a brush object.

I Input Field—Adds an input field.

W Text Window—Adds a text window.

Database Editor Keyboard Shortcuts

***A* I** Insert—Inserts the current record in the database.

***A* U** Update—Updates the database with the new information for the current record.

***A* D** Delete—Deletes the current record from the database.

***A* C** Clear—Clears all the data in the current record without affecting the file.

***A* N** Next Record—Jumps to the next record if there is one.

***A* P** Previous Record—Jumps to the previous record if there is one.

***A* ,** Previous Page—Jumps to the previous page of data if there is one.

***A* .** Next Page—Jumps to the next page of data if there is one.

***A* R** Record Number—Toggles between advancing records by Key Field or by Record Number.

AAAE VIDEO PLAYER DEVICE C/ASSEMBLER PROGRAMMER'S REFERENCE

Documentation and software by Hanafi Houbart

Ariadne Software Ltd.
273 Kensal Road,
LONDON
W10 5DB

Introduction

This document is intended as a programmer and designer reference for the system level of the Ariadne Video Player Device (AVPD). This appendix deals with the interface that programmers working from C and Assembler would use with the player device.

In dealing with the programming aspects of the AVPD, it is assumed that the reader is familiar with the installation of the AVPD and the basic facilities offered by the device as accessed using the PlayerControl utility.

The interface has been used with Lattice 3.10 and 4.01, and with the MetaComCo Assembler and is entirely compatible with both compilers and assembler.

The device can be used *synchronously,* whereby each I/O call to the device will hold control in the programmer's code until the I/O on that call is completed. Synchronous I/O is generally the most useful mode for controlling video players. *Asynchronous* operation allows control to return immediately to the caller, who must elect to 'wait' or 'check' on the completion of the command call. Time critical or specialized applications may well wish to use this method of working.

This appendix assumes you can understand the C programming language.

The Video Player Device

The advanced player device is an asynchronous 'Exec' type device driver, normally kept in the **DEVS:** directory of the boot media.

The name of the device is **'PLAYER.DEVICE'.**

The player.device follows all of the conventions for Exec type devices outlined in the *ROM Kernel* manuals. The device is kept on disk until such type as a process opens the device using the standard *OpenDevice( )* Exec call.

Player.device is a fully asynchronous device, in that it creates a separate I/O task under Amiga multitasking to process player communications.

Player.device expects *exclusive* access to the low-level resource 'serial.device', and in an Amiga with unexpanded serial ports, it will require exclusive use of the on-board serial port.

In order to support the various player protocols, the device loads a 'translation' driver that converts generalized player requests into specific signals that the selected player can understand. The driver file is used to configure the serial interface to the most convenient mode for the player in question. Baud rates, handshakes, etc., are fixed at their optimum settings.

Once the player.device is loaded and initialized, player communication can commence. Normally the device will require a request to perform 'soft' initialization to operate correctly. This process is documented below.

The process for communicating with player.device is basically an extension of the standard I/O protocol for any Exec device outlined in the *ROM Kernel* manuals. Requests are sent to the device using a modified *I/ORequest* structure.

Drivers

The device refers to a subdirectory **'DEVS:PLAYERS'** in which the device expects to find video player specific device driver files.

Implemented player drivers are:

Philips_405_1200
Philips_410_9600
Philips_835_1200
Pioneer_2200_4800
Pioneer_4200_4800
Pioneer_6000
Sony_1200_9600
Sony_1500_1200
Sony_1500_9600
Sony_1550_1250
Sony_1550_9600
Sony_2000_9600
Sony_Umatic9_9600

Drivers are named using a **MAKE_MODEL_BAUD** convention in the driver's filename.

The driver selection is rendered visible to player.device through a simple, one line, ASCII text file containing the filename of the selected driver called **DEVS:Player-Configuration**. Programmers who wish to create their own utility to select drivers merely need to Open/Write a new filename into this file before invoking the device.

The device keeps an internal record of how many times it has been opened by Exec. Exec will expunge the device and driver if all processes using the device legally close it by calling *Exec Close Device( )* and *Rem Device( ).* The driver will be reloaded on the next open.

If the DOS Layer has been installed, the DOS message handler will have a permanent Open held on the device, consequently an old driver will never be discarded under normal use until the machine is rebooted.

If the programmer must discard and load alternative driver files 'on-the-fly,' commands exist in the generalized command set to do this. Discard/reloads will only function when changing drivers between similar models of video players; a change of make will require the device to be closed.

Programmers who must change the RS232 settings of a given driver can 'hack' an existing driver. Programmers that do so can expect no support from Ariadne in the performance of a hacked driver. Drivers consist of a TAG header and the driver code itself. The Tag contains all the data the device requires to identify the driver and set the RS-232 settings. The structure is known as a *PlayerTag* structure and always occurs at the start of the driver file:

```
  /* driver header structure - [from Player.h] */
struct PlayerTag
char   *playername;           /* driver name */
int    (*Init) ( );           /* initialization routine */
int    (*Expunge) ( );        /* expunge routine (if present) */
int    (*Service) ( );        /* pointer to driver main */
int    (*ExtCad) ( );         /* pointer to device I/O patch */
int    VertBValue;            /* default dead time count */
int    TermChar;              /* char to terminate write */
int    playerID;              /* driver specific identity code */
```

```
LONG   TAG_I/OBUFLEN;       /* length of internal I/O buffers */
LONG   TAG_I/OBAUD;         /* baud rate rs232 */
LONG   TAG_I/OBRKTIME;      /* breaktime rs232 */
LONG   TAG_I/OTERM0;        /* terminators */
LONG   TAG_I/OTERM1;

BYTE   TAG_I/OREADLEN;      /* read length bits */
BYTE   TAG_I/OWRITELEN;     /* write length bits */
BYTE   TAG_I/OSTOPBITS;     /* # stop bits */
BYTE   TAG_I/OEOFFLAG;      /* parsing eofs on cad flag */
BYTE   TAG_I/OSERF7;        /* 7/3 wire flag */
BYTE   TAG_TERMFLAG;        /* include termchar on write flag */

LONG   Ident;               /* driver ID marker 'AAPD' */
```

Accessing The Device

The device is accessed via a call to Exec *OpenDevice( )*. Prior to this call, a port and I/O structure must be created for the communication to proceed.

The port is created in the standard way:

```
    .
    .
    if ( (port = CreatePort(0,0) ) = NULL) Exit(1);
    .
    .
```

The I/O structure is an extension of the Amiga standard I/O request structures *I/ORequest/I/OStdReq*, called an *I/OPlayer* structure:

In the outline that follows, 'reserved' objects are reserved for use by ASL; using these objects will create conflicts with future implementations.

```
struct I/OPlayer /* from Player.h] */
{
  /* standard Request components... */
  struct Messageio_Message; /* I/O standard */
  struct Device  *io_Device; /* I/O standard */
  struct Unit    *io_Unit;   /* I/O standard RESERVED */
  UWORD          io_Command; /* AC_Code value */
  UBYTE          io_Flags;   /* RESERVED for AVPD use ONLY! */
  BYTE           io_Error;   /* error return */
  ULONG          io_Actual;  /* internal/system calls only */
  ULONG          io_Length;  /* internal/system calls only */
  APTR           io_Data;    /* internal/system calls only */
  /* player extension components... */
  UWORD          iop_Format; /* FORMAT_value */
  APTR           iop_Argin;  /* pointer to input */
  APTR           iop_Argout; /* pointer to output */
  LONG           iop_Argx;   /* general parameter in/out */
  LONG           iop_Argy;   /* general parameter in/out */
  UWORD          iop_Macro;  /* Macro parameter in/out */
{;
```

This and other constructs required in player I/O are supplied in files Player[.h/.i].

Typically the programmer using synchronous I/O will create a single I/OPlayer block for all player communications in the DATA area of his or her code.

In asynchronous I/O an arbitrary number of such blocks may be dynamically allocated.

```
.
.
if ((I/OBlock = (struct I/OPlayer #)
CreateExtI/O(port,sizeof(struct I/OPlayer))  ) = NULL)
{
   printf(''Cannot create I/OBLOCK - Aborting!!!\n\n'');
   goto cleanupl;
};
.
.
```

Inexperienced programmers are reminded that I/O blocks of all kinds (including *I/OPlayer*) become the exclusive 'property' of the I/O Device once an I/O request has been initiated, and remain so until the device signals that I/O on that block is completed.

Having created an *I/OPlayer* structure, it must be passed to the device for initialization on device open:

```
/* Where I/OBlock is a pointer to struct I/OPlayer */
.
error = OpenDevice(PLAYER_DEVNAME,0,I/OBlock,0);
.
```

PLAYER_DEVNAME is defined in the include files.

The returned structure is initialized for use with the player.device.

All other I/OPlayer structures used with the device must be 'cloned' from this first initialized structure:

```
*NewBlock=*OldBlock;
```

The initialized I/O blocks are loaded with appropriate data and passed to the device through the Exec I/O series functions: *DoI/O( )* for synchronous operation and *SendI/O( )* for asynchronous.

Upon opening the device, the following actions are taken:

All internal functions required by the device are initialized EXCEPT the device task.

A call is made to the resident driver code to load and initialize the specific device driver.

The driver's server first releases any existing driver, Macros and allocated memory, then loads the new driver into memory. The driver is checked for an identifying LONG word "AAPD," and the driver is discarded if this ID is not found.

If all is well the device then initializes its task, and the device is ready for use.

Upon successful opening the device initializes the *I/OPlayer* structure (presented through the call to Open) and returns a pointer to the driver name in *I/OPlayer*->**iop_Argout.**

The device name, etc., can be retrieved through the Device pointer returned in the initialized *I/OPlayer* structure.

```
/* Where I/OBlock is a pointer to an initialized I/OPlayer
structure */
.
PlayerBase = (struct Library #) (I/OBlock->io_Device);
printf( ''DEVICE ADDRESS = %OBIX\n'' , PlayerBase );
printf( ''DEVICE NAME = %s\n'', PlayerBase->lib_IdString );
.
.
```

The initialized block contains the address, etc., of the device to which it 'belongs.' It is this kind of data that must be copied into any additional *I/OPlayer* structures you wish to use.

The integrity of the I/OBlock must be preserved for the duration of player communications.

The I/OPlayer Structure

The structure member, io_Flags, is reserved for internal use by the device.

The structure members io_Actual, io_Length, and io_Data are for internal use by the device in communicating with the player and for programmers who wish to send commands directly to the player without the intervention of the device.

Parameters are passed TO the device using the members :
iop_Argin, iop_Argx, iop_Argy and iop_Macro.
Data, where appropriate is returned FROM the device in :
iop_Argout, iop_Argx, iop_Argy, io_Error.

The iop_Format member is used, where appropriate, to indicate the format of the supplied parameters.

Parameters may be optionally of type integer (passed in iop_Argx and iop_Argy) or of type string pointed to from iop_Argin, which must be NULL terminated and no longer than 32 bytes long including NULL *except when the string pointed to is a valid AmigaDOS file name.*

Structure member io_Unit is currently unused but supported as a standard member of an *I/ORequest* structure *and will be used at a later time by ASL.* This should be set to NULL to ensure upward compatibility.

Direct and Device I/O:

io_Command — Exec/ACode number for the command.
io_Error — Zero on return or Error number

Direct I/O:

io_Data — Points to input/output buffer
io_Length — Size of read/write to perform in bytes
io_Actual — Size *actually* read on 'read' in bytes

Device I/O:

iop_Argx/Argy — Integer parameters
iop_Argin — Pointer to data in
iop_Argout — Pointer to data out
iop_Macro — Integer Macro ID

The Command Set

Commands are passed to the device by setting the Command member of the *I/OPlayer* structure to a particular value.

The command set, outlined in the appendix, is generalized and applies to all players supported by the system.

The commands fall into three main categories:

Device Commands

Device commands affect the configuration of the player device and its attendant device driver. The device may be locked to a given set of operational parameters, ensuring application security.

Optional levels of error reporting may be selected (and locked), the current driver discarded and/or replaced, and the device interrogated for information about the video media currently loaded.

ATOMic Commands

These are a wide selection of single, simple actions, most video players can be called upon to perform. Each specific player driver will action the necessary communication protocol to cause each atom event to happen.

A comprehensive list (as currently implemented) is attached to this document outlining the function of each ATOM call.

Macro Commands

Macro Commands are user-defined aggregates of DEVICE, ATOM and MACRO calls that perform a compound player function. MACROs are identifiable by either a unique integer number or by use of ASCII null terminated text string.

The device will support a maximum of $7FFF macros within the limitation of available memory. Macro definitions may be saved, reloaded and/or merged from computer media storage.

The uses of these commands are outlined below.

Commands are identified by their Ariadne player device Code or ACode.

ACodes are listed at the end of this appendix and defined in the files PlayerCode[.h/.i] for application use. These files also provide commented identifiers for the error codes the device can return.

Device-reserved commands are those supported by all Exec devices and include such functions as device abort and flush, etc. These are available for programmer use if the programmer requires direct communication with the player and for downward compatibility with earlier ASL devices.

Communicating directly with the player through device-level calls is supported, but the Player Device may lose track of the state of the player. The commands **AC_DCLEAR** and **AC_ARCLRR** should revive even the most confused device after direct access.

Invalid commands are commands that cannot be supported by the device and are (effectively) any command whose number is greater than $AO and unavailable command numbers.

Unavailable commands are commands which should exist but don't, either because they haven't been defined, written or because they can't be (e.g., instant jump on early Sony players).

Unimplemented commands are commands, in the virtual set, whose implementation is not possible for a given video player and whose absence is not regarded as critical—e.g., video player beep. In this case no error is generated nor any action taken, but a warning code is returned to the application in **iop_Argx.**

ATOM calls request the device to perform some specific action:

Parameters are supplied either as **strings** or as **integers.**

iop_Argx always contains the video address if the command requires one.

iop_Argy usually contains a modifier if required.

Upon completion of an I/O request, an error may be returned by the device (whose meaning is outlined in the appendix and the supplied include files.)

In the event that no driver has been loaded or the driver is absent, ALL ATOMic commands will return an error.

Device-internal I/O may be of two types: Synchronous and Asynchronous.

Some default commands and some device-modifying commands are always executed synchronously, otherwise a call to the device's support task is made. In both cases the resultant execution path is largely identical.

Direct access calls are processed synchronously in the device by low-level read/write functions.

Commands with codes greater than **AC_CREADY** ($09) are vectored through the device's high-level asynchronous server (with the exceptions of Macro Load and Save).

Commands whose actions require servicing in the device are handled by this server. This includes macros, device/driver locking and commands that affect the configuration of the driver.

Only valid specific ATOMs are passed on to the driver.

Extensive return checking is done on all RS232 access to check for unpredictable process messages coming from the player. When detected, these results are stored in data and then acted on on the next return of an I/O block. Player commands that can solicit an immediate error from the player are checked for success, and an appropriate error is returned on failure (e.g., disc tray open).

If the player requires any significant overhead in processing an action request, the device simply lets the player get on with it and checks on subsequent programmer calls to the device, whether or not the player has responded.

If a result exists for return to the calling process, the appropriate fields are set up:

> If an error has occurred it is entered into the io_Error field of the *I/OPlayer* structure, and the same code is returned to the device server.
>
> In the event of an Error, the field iop_Argout will contain a pointer to a NULL terminated string describing the nature of the error in English.

This description is both informal and incomplete and leaves aside the variations between individual commands.

Macros

The intention behind the method of macro implementation used is to allow a calling application minimum difficulty in creating and controlling macro commands.

To create a Macro, the device requires a valid Macro identity to be declared, thus signalling the intention to define a Macro. Thereafter valid ATOM calls sent to the device—rather than being immediately executed—are stored internally in a linked aggregate until such time as an end of definition call is issued by the application.

The command aggregate can then be called by issuing a Macro call identifying the Macro either by number or by name.

There are limitations placed on the ATOMs permissible in a Macro aggregate:

1. Macros may not issue calls to themselves (to prevent recursion).
2. Macros may not issue calls to undefined Macros (to prevent recursive loops).

3. Macros may not Create, Load, Save or Merge Macros or macro files.
4. Macros may not reconfigure the device.

All these conditions are dynamically error checked during macro definition.

Additionally, macros are nestable up to 32 levels deep. Nesting overflow is only detected on macro execution. This level is arbitrary and may be modified in later versions if the need becomes apparent.

The macro driver allows the user to specify a given Macro number or 'ID' to be defined by the user in which case a macro textual 'name' is optional (but useful).

Alternatively, the user may define only the **'name'** of the new macro and allow the device to sort out macro numbering. This has the benefits of being more process efficient and allowing a utility, such as a macro editor, to 'delete the last entered' / 'create and next' macros automatically. This is the mode used in the *Command Text Interpreter* and *PlayerControl.*

The macro facility is very powerful. Complex video sequences and player command composites (e.g., video on & audio 1 on & audio 2 off) may be assembled outside the execution of an application and saved to disc. The final application may then be authored using commands that refer to meaningful concept names (e.g., *'Introduction_2'*, *'Player_Ready'*, etc.) rather than cryptic bundles of low-level calls. This is both friendly and memory/execution efficient.

Macro names may consist of *any* non-null ASCII characters when using the device at system level.

Programming Criteria

Certain facilities of the device must be used in order to gain maximum benefit. Upon starting an application a designer should always issue an AC_AREADY call.

This will cause the player to INITialize itself and enable the device driver to intelligently access the media it has been presented with.

Frame and time media are discerned and commands excluded either by media format or player capability (e.g., instant jump on a long-playing video disc); they are automatically switched out of the driver. In its default state the driver will return an I/O error if given a disc non-compatible command. Applications that require these errors to be suppressed may issue an error escape call.

The sound format of the disc is assessed and the audio channels configured to match. Stereo media have both audio channels enabled, bilingual, mono or silent media audio one only.

Upon successful completion, AC_AREADY returns a string pointer in Argout that usefully describes the disc format, for example:

```
[LASERDISC : FRAMECODED : BILINGUAL]

[TAPE : TIMECODED : MONO]
```

Later in this document is a sample program that illustrates the AVPD used from C in a simple synchronous demo. This illustrates a convenient way of accessing the player device.

Appendix A contains a brief description of each command and its effect.

It is not necessary—but can be useful—to issue a device CLEAR call (AC_DCLEAR) before each I/O operation. This may result in uncleared messages being lost to the device.

Programming [illegible]

[illegible]

[illegible]

The string to be sent should be NULL, terminated in the usual C convention; if this is done, the device can work out how long the string is:

```
/* Where I/OBlock is a global I/OPlayer pointer.
*/

int WriteText (text)
char *text;
{
  I/OBlock->io_Command = AC_DWRITE;
  I/OBlock->io_Length = -1;
  I/OBlock->io_Data = (APTR) text;

/* The value from DoI/O = error number or ZERO */

return ((int) DoI/O(I/OBlock));
}
```

If the programmer wishes to specify an exact length in io_Length, he or she may do so.

Direct Read

Direct read is achieved using the inverse of the above process. The *I/OTERM* members of the driver Tag determine which characters will be regarded as terminating input when −1 is specified as the read length; otherwise control is hung until io_Length characters have been received from the player.

Of much more use to the direct access programmer is the *Conditional* read function. This allows programmers to specify how many bytes they expect to get (or any number with −1), and the call will return immediately with as many bytes up to the limit as could be read and transferred to the io_Data buffer. The number of bytes actually read or ZERO will be in io_Actual:

```
/* Where I/OBlock is a global I/OPlayer pointer */

int MayRead ()
{
  I/OBlock->io_Command = AC_DCREAD;
  I/OBlock->io_Length = -1;
  I/OBlock->io_Data = (APTR) AGlobalBuffer;

  DoI/O(I/OBlock);

  /* io_Actual = the number of bytes read or ZERO */

return(I/OBlock)->io_Actual);
}
```

This function can be repetitively called until a reply is received or called as a general catch-all to flush any sporadic replies from the player before starting a new write.

The other device standard calls perform as documented in *ROM Kernel*, but are generally of little use to the application programmer.

Device Access

The symbol AC_xxxxxx is used to indicate any valid command code supported by the player. These AC_Codes are defined in supplied files PlayerCode[.h/.i]. The commands are sub-divided into several categories that reflect the general method used when invoking the command.

Write Commands

Commands of type *WRITE* require only the **Command** field of the I/OBlock to be set up:

```
.
.
I/OBlock->io_Command=AC_xxxxxx;
DoI/O(I/OBlock);
.
```

The Command may return data in the form of integers or string pointers according to the specific command function.

Read Commands

Commands of type *READ* are commands that explicitly return data in the form a string with the resultant read returned as a pointer in **iop_Argout.**

```
.
.
I/OBlock->io_Command=AC_xxxxxx;
DoI/O(I/OBlock);
printf(''returned %s\n'',(char *) I/OBlock->iop_Argout);
.
```

ARGSX(Y) Commands

Commands of type ARGSX (ARGSXY) require x (and y) parameters or alternatively a string:

To goto and stop at frame 12345 (using FORMAT_ARGS).....

```
    .
    .
    I/OBlock->io_Command=AC_AFGOST;
    I/OBlock->iop_Format=FORMAT_ARGS;
    I/OBlock->iop_Argx=12345;
    DoI/O(I/OBlock);
    .
```

(Using FORMAT_STRING with a Philips player).....

```
.
.
I/OBlock->io_Command=AC_AFGOST;
I/OBlock->iop_Format=FORMAT_STRING;
I/OBlock->iop_Argin=''12345'';
DoI/O(I/OBlock);
.
```

IMPORTANT!

It should be noted that the *arguments* method is player-independent and that the *string* method is not due to the requirement to syntactically construct the command string from the application. The string parameter facility has been included to ease application development where both the player is known to be of a particular type and the control vehicle most conveniently expressed as a string.

Applicaton workers should, wherever possible, use the argument method of player control, as this is guaranteed to be player independent.

The above command performs a SEARCH and must be checked for completion using AC_AQENDM (= = WHEN PLAY in DOS/Mtext).

MACRO Commands

Macro calls are extensions of type ARGSX where the **iop_Macro** field is used to contain a valid Macro ID number.

When usng macro calls the ID number should be entered into **iop_Macro** and/or the macro name in **iop_Argin**.

```
  /* return error as function value */

  return(ret);
}
LONG CommandXY(acode,x,y,text)
int acode,x,y;
char *text;
}
  /* set parameter from function into I/O block
*/

  I/OBlock->iop_Argx = x;
  I/OBlock->iop_Argy = y;
  I/OBlock->iop_Argin = (APTR) text;

  /* call CommandAtom passing command number
    on.. */
}
```

Using the above functions, a call to FIND 12345 would appear as follows:

```
CommandXY(AC_AFGOST,12345,0,NULL);
```

Similarly, to start a macro:

```
CommandXY(AC_MADDLC,0,0,''Eric_the_low-
  level_Macro'');
```

Constructs using state tables and other higher level structures can be built upon this kind of function. After each call the global *I/OPlayer* can be accessed to collect returned values:

..to read current frame code..

```
CommandAtom(AC_AFREQ);
currentpos=atoi(I/OBLOCK->iop_Argout);
```

ACode Command Reference

Direct Access Commands AC_Direct

AC_D . . . . $0n Device control codes

AC_DINVLD $00 DIRECT Invalid Command

This Command causes an error.

AC_DRESET $01 DIRECT Reset I/O Device

Should not be called by an application unless direct access requires it.

Corresponds to Amiga exec RESET call.

AC_DREAD $02 DIRECT Direct I/O READ

Should not be called by an application unless direct access requires it.

Corresponds to Amiga exec READ call.

AC_DWRITE $03 DIRECT Direct I/O WRITE

Should not be called by an application unless direct access requires it.

Corresponds to Amiga exec WRITE call.

If called with NULL io_Data pointer will cause to device to wait iop_Argx number vertical blanks.

AC_DFLUSH $04 DIRECT Force Update I/O buffers

Should not be called by an application as no useful purpose is served.

Corresponds to Amiga exec FLUSH call.

AC_DCLEAR $05 DIRECT Clear all I/O

Clear internal buffers.

Applications may call this to ensure the device is ready.

Corresponds to Amiga exec CLEAR call.

AC_DSTOP $06 DIRECT Stop I/O

Should not be called by an application as no useful purpose is served.

Corresponds to Amiga exec STOP call.

AC_DSTART $07 DIRECT Start I/O

Should not be called by an application as no useful purpose is served.

Corresponds to Amiga exec START call.

AC_DABORT $08 DIRECT Abort All pending I/O

Applications may call to stop the device dead and clear out any I/O.

Corresponds to Amiga exec ABORT call.

CAVEAT! This function will only abort uncleared I/O, if the player has already begun an action, the player will probably complete it. This call is best reserved for getting out of 'stuck' devices rather than being used as a general method of control.

AC_DCREAD $09 DIRECT Conditional I/O Read

Should not be called by an application unless direct access requires it.

Checks RS232 buffer for input—if any found returns as exec READ with number of bytes read in io_Actual else io_Actual returns zero.

Application Device Codes AC_Device

AC_D . . . . $0n Device application codes

AC_DPLAYR $0A READ Query player selected

Arguments in: NONE

Arguments out: Argout—Driver ID string Pointer or NULL.

Special errors: Excluded if macro parsing.

AC_DTYPE $0B ARGSX Set Error escapes

Arguments in: Argx—Error escape flags (PlayerCode.i/h)

Arguments out: NONE

Special errors: Excluded if locked or macro parsing.

AC_DLOCK $0C WRITE Lock device configuration

Special Errors: Excluded if macro parsing.

AC_DULOCK $0D WRITE Unlock configuration

Special Errors: Excluded if macro parsing.

AC_DQUERY $0E READ Query parse mode

Query if device is active or parsing a macro

Arguments in: NONE

Arguments out: Argx—True if Macro parsing.

AC_DVECT $0F ARGSXY Special needs vector

Arguments in: Argin—Pointer to write string
Argx—# bytes to read from player (0-511)

Arguments out: Argout—Pointer to reply buffer or NULL

Special error: "Error in input parameters"

Player Status Codes AC_Player

AC_P ....$1n Player status/select query

AC_PSELCT $10 ARGSX Load named driver

Arguments in: Argin—Pointer to driver name
EXCLUDING directory path

Arguments out: NONE

Special errors: Excluded if locked or macro parsing.
"Cannot access driver file",
"Driver load failed",
"Nonexistent driver requested",

AC_PQUERY $11 READ Enquire driver name

Enquire player driver currently loaded
Arguments in: NONE

Arguments out: Argout—pointer to player name.

AC_PIDENT $12 ----- Enquire player on RS-232

NOT IMPREMENTED!

AC_PUSLCT $13 WRITE Discard current driver

No Parameters—forcibly frees driver memory if resident.
THE DEVICE WILL NOT FUNCTION UNTIL A NEW DRIVER IS LOADED!

AC_PNULL $14 - $1F reserved

Special Functions AC_eXtension

AC_X . . . . $2n Special device functions

AC_XVWAIT $20 ARGSX Wait on Vertical Blank

Arguments in: Argx number of FRAMES to wait

Arguments out: none

CAVEAT! Frames = 2x Vertical Blanks

AC_XNULL $21 $2F reserved

Macro Calls AC_Macro

AC_M. . . . . $3n User Macro Allocation calls

AC_MRESET $30 WRITE Reset Macro List

Reset macro list. Removes ALL currently resident Macros and frees any memory used thereby.

No Parameters

Special errors: Excluded if locked or Macro parsing

AC_MALLOC $31 MACROXY Start Macro Allocation [n]

Allocate a new macro to **ID** n—*name is optional.*

Arguments in: Macro—macro ID to allocate
Argin—pointer to optional name or NULL

Arguments out: Argout—name echoed back
Macro—ID actually allocated to.

Special errors: Excluded if locked or Macro edit locked

(see errors) Also name, recursion and ID errors trapped

AC_MADDLC $32 MACRO Start Allocation Next ID

Start Macro Allocation next free ID, highest ID plus one is used—gaps in macro list will NOT be filled.

Arguments in:	Argin—pointer to unique name
Arguments out:	Argout—name echoed back Macro—ID actually allocated to.
Special errors: (see errors)	Excluded if locked or Macro parsing Also name, recursion and ID errors trapped

---

AC_MENDLC $33 READ End Macro Allocation

Clost Macro Allocation

Arguments in:	NONE
Arguments out:	Argx—ID of macro closed (or -ve if not allocating) Argy—number of atoms in macro (or zero) Argout—pointer to name (or NULL)
Special errors:	As above

NOTE! If no allocating no error will be caused, hence the is command can be used for saftey if required.

AC_MFREEN $34 MACRO Delete Macro by ID/Name

Arguments in:	Macro—ID of Macro AND/OR Argin—name of macro

ID number will be used if given, name otherwise.

Note: Define value NOMACRO = = null identity

Arguments out:	NONE
Special errors:	As above

AC_MFREEL $35 MACRO Delete last Macro in List

No Parameters.	deletes last macro allocated by AC_MADDLC
Special errors:	As above

AC_MCALLN $36 MACRO Call Macro by Name

Arguments in: Argin—pointer to macro name

Arguments out: Argout—pointer to name of macro called
Argx—ID of macro executed
Argy—number of atoms executed

Special errors: as above but also..

"No valid macro identity"—macro does not exist

In the event of an error in a nested macro, Argx will contain the ID number of the macro that failed.

AC_MCALLS $37 MACRO Call Macro by ID

Arguments in: Macro—macro ID number.

Arguments out: Argout—pointer to name of macro called
Argx—ID of macro executed
Argy—number of atoms executed

Special errors: as above but also..

"No valid macro identity"—macro does not exist.

In the event of an error in a nested macro, Argx will contain the ID number of the macro that failed.

AC_MLOAD $38 ARGSX Load Macro Set

Arguments in: Argin—pointer to macro file name
EXCLUDING directory

Arguments out: Argout—pointer to macro set name

Special errors:

Usual filling errors, also the macro file is checked and unpacked during load, all of which can fail.
Will be excluded if device locked.
See also merge/edit locks below.
CAVEAT! the directory to load macros from defaults to :
DEVS:Players/Macros/
See call AC_MDIR to change filing directory.

AC_MSAVE $39 ARGSX Save current Macro Set
Arguments in: Argin—pointer to save name EXCLUDING Directory path.
Arguments out: None
Special errors: There must be something to save!

AC_MEDIT $3A ARGSX Set edit configuration
Arguments in: Argx—Macro edit flags (see PlayerCode.h/.i)
Arguments out: NONE
Special errors: Excluded if device locked.

AC_MNAME $3B ARGSX Identify Macro by ID
Arguments in: Macro—Macro ID or NOMACRO
Arguments out: Argout—pointer to macro name or NULL
Argx—total number of macros held by device
Argy—number of atoms in macro x or 0
If called with ID = = NOMACRO total macro count only returned.
If macro non-existent for slot NULL name returned.

AC_MSTOP $3D WRITE Suspend Allocation
Arguments in: None
Arguments out: Argx—TRUE = stop, FALSE = not allocating

AC_MSTART $3E WRITE Restart Allocation
Arguments in: None
Arguments out: Argx— = NOMACRO if not previously stopped else same as END ALLOCATION (AC_MENDLC)

AC_MDIR $3F WRITE Set Macro filing pathway
Arguments in: Argin—Pointer to Directory pathname
Arguments out: NONE
Special errors: Excluded if device or macros locked.

Atoms AC_Atoms

AC_A . . . . $4n+ Player Atoms

AC_AR. . . . . ATOM Reset Codes:

AC_ARESET. . . . . . . $40 WRITE Reset Player

Player reset to power-on condition.

No parameters.

AC_READY $41 WRITE INITIALSE PLAYER

Player to ready state

MUST be called when first opening device.

Arguments in:	NONE
Arguments out:	Argout—pointer to a string that describes the video media.

This function places the player into it's ready state and intialises the AVPD internal disc flags and switches out commands that will not work with the current disc format.

TAPE/CAV discs :—intialise to first frame with video ON

CLV discs :—plays from start of disc.

AUDIO :—Stereo—both channels ON else channel 1 ON only

AC_AROFF $42 WRITE Player to idle

Player to standby state

No parameters. Video off—disc stationary.

AC_ARCLRR $43 WRITE Discard internal registers

Throw internal registers away

Internal STOP and REPORT registers discarded.

No Parameters. May cause player to STOP.

AC_AH .... Atom Hardware Controls:

AC_AHVINT	$46	WRITE	Video internal
AC_AHVEXT	$47	WRITE	Video external
AC_AHJDIS	$48	WRITE	Eject disable
AC_AHJENB	$49	WRITE	Eject enable
AC_AHJECT	$4A	WRITE	Eject disc
AC_AHRENB	$4B	WRITE	Replay enable
AC_AHRDIS	$4C	WRITE	Replay disable

AC_AHMON $4D ARGSXY Philips RC-5
Monitor command (Philips RC-5 only)
Arguments in: Argx—
Argy—refer to Philips documentation.

AC_AHUSER $4E ARGSXY Philips Control
Configure user accessed controls (Philips only)
Arguments in: Argx—
Argy—refer to Philips documentation.

AC_AP............ Atom Player Query:

AC_APLOAD $50 READ Query disc loaded
Arguments in: NONE
Arguments out: Argx—TRUE if condition true

AC_APRPLY $51 READ Query replay hardware enabled
Arguments in: NONE
Arguments out: Argx—TRUE if condition true
(player dependent)

AC_APUSER $52 READ Query user controls enabled
Arguments in: NONE
Arguments out: Argx—TRUE if condition true

AC_APROM $53 READ Read player ID/ROM code
Arguments in: NONE
Arguments out: Argout—pointer to ROM/VERSION string

AC_APSTAT $54 READ Read player status string
Arguments in: NONE
Arguments out: Argout—pointer to player status descriptor

AC_APFAST $55 READ Query FAST speed limit
Arguments in: NONE
Arguments out: Argx—0 or maximum speed multiplier

AC_APSLOW $56 READ Query SLOW speed limit
Arguments in: NONE
Arguments out: Argx—0 or maximum speed divisor

AC_AQ .... Disc Query:

AC_AQTYPE $58 READ Query media coding
Arguments in: NONE
Arguments out: Argx—TRUE if FRAME/FALSE if TIME

AC_AQSTAT $59 READ Program status request
Arguments in: NONE
Arguments out: Argout—points to status request string

AC_AQUSER $5A READ User code request
Arguments in: NONE
Arguments out: Argout—points to User Code string

AC_AQENDM $5B WRITE Query End of Search Phase
Arguments in: NONE
returns when condition TRUE = WHEN PLAY in DOS Layer

AC_AA............	Audio Codes:		
AC_AABEEP	$60	ARGSXY	Sound Insert (beep)
Arguments in:	Argx/Argy—pitch/duration		
(player dependent)			
Arguments out:	NONE		
AC_AAMUTE	$61	ARGSX	Audio mute over non-play phase
Arguments in:	Argx—TRUE is On, FALSE Off.		
Arguments out:	NONE		
AC_AACX	$62	ARGSX	CX On/Off
Arguments in:	Argx—TRUE is On, FALSE Off.		
Arguments out:	NONE		
AC_AA1ON	$63	WRITE	Audio 1 ON
AC_AA1OFF	$64	WRITE	Audio 1 OFF
AC_AA2ON	$65	WRITE	Audio 2 ON
AC_AA2OFF	$66	WRITE	Audio 2 OFF
AC_AAUDEX	$67	ARGSX	Audio Internal/External
Arguments in:	Argx—TRUE is External, FALSE Internal.		
Arguments out:	NONE		
AC_AV............	Video Codes:		
AC_AVPION	$68	WRITE	Picture Index ON
AC_AVPIOF	$69	WRITE	Picture Index OFF
AC_AVCION	$6A	WRITE	Chapter Index ON
AC_AVCIOF	$6B	WRITE	Chapter Index OFF
AC_AVVION	$6C	WRITE	Video ON
AC_AVVIOF	$6D	WRITE	Video OFF
AC_AVOVLY	$6E	ARGSXY	Set/Read Overlay processor
Set/Read Video processor	Refer to Philips documentation.		
Arguments in:	Argx/Argy		

AC_AVTXT $6F ARGSX Teletext ON/OFF
Arguments in: Argx—TRUE is On, FALSE Off.
Arguments out: NONE

AC_AF .... Frame Codes:

AC_AFREQ $70 READ Frame code request
Arguments in: NONE
Arguments out: Argout—pointer to returned string
= "99999" if FRAME read not supported

AC_AFINFO $71 ARGSX Load INFO register
Arguments in: Argx—Frame to REPORT on
Arguments out: Argout—acknowledgement
The device MAY lock until REPORT frame occurs
(player dependent)

AC_AFSTOP $72 ARGSX Load STOP register
Arguments in: Argx—Frame to STOP on
Arguments out: Argout—acknowledgement
The device MAY lock until STOP frame occurs at which point the video will still.
(player dependent)

AC_AFGOST $73 ARGSX Goto FRAME and still
Arguments in: Argx—Frame to search to
Arguments out: Argout—acknowledgement

AC_AFGOPL $74 ARGSX Goto FRAME and play
Goto FRAME and play
Arguments in: Argx—Frame to search to
Arguments out: Argout—acknowledgement

AC_AFGOCN $75 ARGSX Goto FRAME and continue
Goto FRAME and continue previous play mode
Arguments in: Argx—Frame to search to
Arguments out: Argout—acknowledgement

AC_AFRETI $76 WRITE Wait until FRAME INFO clears

AC_AFRETS $77 WRITE Wait until FRAME STOP clears

AC_AC............Chapter Codes:

AC_ACREQ $78 READ Chapter request
Arguments in: NONE
Arguments out: Argout—pointer to string
= "999" if Chapter read unsupported

AC_ACGOST $79 ARGSX Goto CHApter and still
As for FRAME

AC_ACGOPL $7A ARGSX Goto CHAPTER and play
As for FRAME

AC_AT............Time Codes:

AC_ATREQ $7C READ Time Code Request
Arguments in: None.
Arguments out: Argout—Time string whose format is..
H:MM'SS Hours Mins Secs
= "9:99'99" if time read not supported

AC_ATINFO $7F ARGSX Load Time code INFO register
Arguments in: Argx—Minutes
Argy—Seconds

AC_ATGOPL $7E ARGSX Goto time and play (CLV)
Arguments in: Argx—Minutes
Argy—Seconds

AC_ATRETI $7D WRITE Wait until TIME INFO clears

AC_AS............Speed Codes:

AC_ASFAST	$80	ARGSX	Set Fast speed modifier
Arguments in:	Argx—speed multiplier value eg 3=3 times normal		
AC_ASSLOW	$81	ARGSX	Set Slow speed modifier
Arguments in:	Argx—speed divisor value eg 3 = 1/3rd normal		
AC_ASSTIL	$82	WRITE	Freeze video
AC_ASSTOP	$83	WRITE	Stop + mute video
AC_AJ	Jump Codes:		
AC_AJUMPF	$86	ARGSX	Instant Jump Fwd
Arguments in:	Argx—frames to jump		
(<=AC_AJUMPQ)	see cmd $88		
AC_AJUMPR	$87	ARGSX	Instant Jump Rev
Arguments in:	Argx—frames to jump		
(<=AC_AJUMPQ)	see cmd $88		
AC_AJUMPQ	$88	READ	Query jumping capability
Arguments in:	None		
Arguments out:	Argx—number frames instant jump or zero		
AC_AJPFJF	$89	ARGSXY	Play Fwd Jump Fwd
Play forward jump forward repetitive			
Arguments in:	Argx—play frame count Argy—jump frame count refer to Philips documentation		
AC_AJPFJR	$8A	ARGSXY	Play Fwd Jump Rev
Play forward jump reverse repetitive			
Arguments in:	Argx—play frame count Argy—jump frame count refer to Philips documentation		

AC_AJPRJF $8B ARGSXY Play Rev Jump Fwd
Play reverse jump forward repetitive
Arguments in: Argx—play frame count
Argy—jump frame count
refer to Philips documentation

AC_AJPRJR $8C ARGSXY Play Rev Jump Rev
Play reverse jump reverse repetitive
Arguments in: Argx—play frame count
Argy—jump frame count
refer to Philips documentation

AC_AJHTJF $8D ARGSXY Halt & Jump Fwd
Arguments in: Argx—halt frame count
Argy—jump frame count
refer to Philips documentation

AC_AJHTJR $8E ARGSXY Halt & Jump Rev
Halt & jump reverse repetitive
Arguments in: Argx—halt frame count
Argy—jump frame count
refer to Philips documentation

AC_AX............Scan Codes:

AC_AXCFWD	$90	WRITE	Scan forward once
AC_AXCRVS	$91	WRITE	Scan reverse once

AC_AM...........Motion Codes:

AC_AMGO	$92	WRITE	Play from media start
AC_AMPNF	$93	WRITE	Play forward NORMAL
AC_AMPNR	$94	WRITE	Play reverse NORMAL
AC_AMPSF	$95	WRITE	Play forward SLOW
AC_AMPSR	$96	WRITE	Play reverse SLOW
AC_AMPFF	$97	WRITE	Play forward FAST
AC_AMPFR	$98	WRITE	Play reverse FAST
AC_AMSTF	$99	WRITE	Step forward 1 frame
AC_AMSTR	$9A	WRITE	Step backward 1 frame

Example Program

```
/***************** Source file PLAYERTEST.C **************** /
  Example Code to test player device
  Hanafi
  © ASL 1987 All rights reserved
  FOR LICENSED USER DOMAIN

 **********************************************************/

*include <exec/types.h>
*include <exec/nodes.h>
*include <exec/lists.h>
*include <exec/ports.h>
*include <exec/libraries.h>
*include <exec/devices.h>
*include <exec/io.h>

*include <stdio.h>

/* NOTE : redirect include as appropriate for your system */

*include ''player.h''

/ *********************************************************/

extern struct MsgPort *CreatePort();
extern struct I/ORequest *CreateExtI/O();

/ *********************************************************/

static struct I/OPlayer *I/Oblock;
static struct MsgPort *port;
static struct Library *PlayerBase;

static char buffer[BUFLEN];

static char *PlayPtr = NULL;
static int MacVal=0;

/ *********************************************************/

VOID CommandPlayer (acode,x,y)
int acode,x,y;
{
  int error;

  I/Oblock->io_Message.nn_Node.ln_Type=NT_MESSAGE;

  printf(''>>>> EXECUTING ATOM  Command=[%02X]\n'',acode);

  I/Oblock->io_Command = acode;
  I/Oblock->iop_Format = FORMAT_ARGS;

  I/Oblock->iop_Argx   = x;
  I/Oblock->iop_Argy   = y;

  I/Oblock->iop_Argin = (APTR) PlayPtr;  /* set this field with
                                          a name */

                                         /* set this field with
  I/Oblock->iop_Macro = MacVal;           a Macro ID */

/* Asynchronous . . . . . . . . . . . . . .

  SendI/O(I/Oblock);
  .
  .
  .
  error = WaitI/O(I/Oblock);

 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .*/

/* Synchronous */

  error = DoI/O(I/Oblock);

  printf(''COMMAND DIAGNOSTICS:command=[%02lX]\n'',acode);

  printf(''ERROR:[%08lX]\n'',error);
  printf(''ARGX:[%08lX]    ARGY:[%08lX]\n'',
    I/Oblock->iop_Argx,I/Oblock->iop_Argy);

  if(I/Oblock->iop_Argout) printf(''<<<< [%s]\n\n'',(char
  *)I/Oblock->iop_Argout);

  getchar();
}

/*********** Main routine **********/

main()
{
  int error;

  if ((port = CreatePort(0,0)) = NULL) Exit(1);

  if ((I/Oblock = (struct I/OPlayer *)
  CreateExtI/O(port,sizeof(struct I/OPlayer))) = NULL)
  {
    printf(''Cannont create I/OBLOCK - Aborting!!!\n\n'');
    goto cleanup1;
  };

  error = OpenDevice(PLAYER_DEVNAME,0,I/Oblock,0);

  PlayerBase = (struct Library *) (I/Oblock->io_Device);

  printf(''\nPLAYER DEVICE DIAGNOSTICS\n OPEN -
  %s\n\n'',(error)?''FAILED'':''** OK **'');
```

```
printf(''ERROR:[%08lX]  ARGX:[%08lX]\n'',error,I/Oblock-
>iop_Argx);

printf(''ARGIN:[%08lX]  ARGOUT:[%08lX]\n'',I/Oblock-
>iop_Argin,I/Oblock->iop_Argout);

printf(''DEVICE:[%08lX]\n'',PlayerBase);

printf(''I/OBLOCK:[%08lX]\n*,I/Oblock);

printf(''\n IDENT:%s'',(PlayerBase)?PlayerBase-
>io_IdString:''-ERROR-\n'');

if(!error)
{
  printf(''DRIVER:%s\n'',(char *)I/Oblock->iop_Argout);
}
else
{
  printf(* ERROR=[ %s ]\n\n'',(char *)I/Oblock-
>iop_Argout);
};

if(error && I/Oblock->io_Device) goto cleanup3;
else if(error) goto cleanup2;

/* OK..assuming FRAMECODED LASERDISC.... */

/* Initialise.. */

CommandPlayer(AC_AREADY,0,0);

/* Just check for frame coded.. */

CommandPlayer(AC_AQTYPE,0,0);

if(!I/Oblock->iop_Argx) printf(''NOT FRAMECODED!\n'');

/* Goto Frame 1000 and stop */

CommandPlayer(AC_AFGOST,1000,0);

/* test for search completed before issuing next command */

CommandPlayer(AC_AQENDM,0,0);

/* OK Play Normal Forward...*/

CommandPlayer(AC_AMPNF,0,0);

/* Wait for 200 frames...*/

CommandPlayer(AC_XVWAIT,200,0);

/* Goto 10000 and fast forward */

CommandPlayer(AC_AFGOCN,10000,0);
CommandPlayer(AC_AQENDM,0,0);
CommandPlayer(AC_AMPFF,0,0);

/* Wait for 200 frames then still...*/
```

```
  CommandPlayer(AC_XVWAIT,200,0);
  CommandPlayer(AC_ASSTIL,0,0);

  /* then off .. */

  CommandPlayer(AC_AROFF,0,0);

  printf(''END OF RUN\n'');
cleanup3:
  RemDevice(I/Oblock->io_Device);
  CloseDevice(I/Oblock);
  printf(''DEVICE CLOSED\n'');
cleanup2:
  DeleteExtI/O(I/Oblock,sizeof(struct I/OPlayer));
  printf(''BLOCK DELTED - '');
cleanup1:
  DeletePort(port);
  printf(''PORT DELETED\n\n'');
  exit(0);
}
/********** End of source file PLAYERTEST.C **********/
```

The above file will compile with Lattice 4 and may be linked using the following response (.with) file:

```
    FROM       LIB:c.o+???:Playertest.o
    TO         ???:PlayerTest
    LIBRARY    LIB:lc.lib+LIB:amiga.lib
```

using appropriate file directories as required.

Player Commands Listed By ACode

AC_D	$0n	Device control code	
AC_DINVLD	$00	DIRECT	Invalid Command
AC_DRESET	$01	DIRECT	Reset I/O Device
AC_DREAD	$02	DIRECT	Direct I/O READ
AC_DWRITE	$03	DIRECT	Direct I/O WRITE
AC_DFLUSH	$04	DIRECT	Force Update I/O buffers
AC_DCLEAR	$05	DIRECT	Clear all I/O
AC_DSTOP	$06	DIRECT	Stop I/O
AC_DSTART	$07	DIRECT	Start I/O
AC_DABORT	$08	DIRECT	Abort All pending I/O
AC_DCREAD	$09	DIRECT	Conditional I/O Read
AC_D	$0n	Device application codes	
AC_DPLAYR	$0A	READ	Query player selected
AC_DTYPE	$0B	ARGSX	Set Error escapes
AC_DLOCK	$0C	WRITE	Lock device configuration
AC_DULOCK	$0D	WRITE	Unlock configuration
AC_DQUERY	$0E	READ	Query parse mode
AC_DVECT	$0F	ARGSXY	Special needs vector
AC_P	$1n	Player status/select/query	
AC_PSELCT	$10	ARGSX	Load named driver
AC_PQUERY	$11	READ	Enquire driver name
AC_PIDENT	$12	----	Enquire player on RS-232
AC_PUSLCT	$13	WRITE	Discard current driver
AC_X	$2n	Special device functions	
AC_XVWAIT	$20	ARGSX	Wait on Vertical Blank
AC_M	$3n	User Macro Allocation calls	
AC_MRESET	$30	WRITE	Reset Macro List
AC_MALLOC	$31	MACROXY	Start Macro Allocation [n]
AC_MADDLC	$32	MACRO	Start Allocation Next ID
AC_MENDLC	$33	READ	End Macro Allocation

AC_MFREEN	$34	MACRO	Delete Macro by ID/Name
AC_MFREEL	$35	MACRO	Delete last Macro in List .
AC_MCALLN	$36	MACRO	Call Macro by Name .....
AC_MCALLS	$37	MACRO	Call Macro by ID ....
AC_MLOAD	$38	ARGSX	Load Macro Set ....
AC_MSAVE	$39	ARGSX	Save current Macro Set
AC_MEDIT	$3A	ARGSX	Set edit configuration ....
AC_MNAME	$3B	ARGSX	Identify Macro by ID .....
AC_MSTOP	$3D	WRITE	Suspend Allocation ......
AC_MSTART	$3E	WRITE	Restart Allocation ....
AC_MDIR	$3F	WRITE	Set Macro filing pathway .
AC_A	$4n +	Player Atoms	
AC_AR	ATOM	Reset Codes:	
AC_ARESET	$40	WRITE	Reset Player ....
AC_AREADY	$41	WRITE	INITIALIZE PLAYER ....
AC_AROFF	$42	WRITE	Player to idle ....
AC_ARCLRR	$43	WRITE	Discard internal registers .
AC_AH.......	Atom Hardware Controls:		
AC_AHVINT	$46	WRITE	Video internal ....
AC_AHVEXT	$47	WRITE	Video external ....
AC_AHJDIS	$48	WRITE	Eject disable ....
AC_AHJENB	$49	WRITE	Eject enable ....
AC_AHJECT	$4A	WRITE	Eject disc ....
AC_AHRENB	$4B	WRITE	Replay enable ....
AC_AHRDIS	$4C	WRITE	Replay disable ....
AC_AHMON	$4D	ARGSXY	Philips RC-5
AC_AHUSER	$4E	ARGSXY	Philips Control ....
AC_AP	Atom Player Query:		
AC_APLOAD	$50	READ	Query disc loaded ....
AC_APRPLY	$51	READ	Query replay hardware enabled ....
AC_APUSER	$52	READ	Query user controls enabled ....

AC_APROM	$53	READ	Read player ID/ROM code
AC_APSTAT	$54	READ	Read player status string
AC_APFAST	$55	READ	Query FAST speed limit
AC_APSLOW	$56	READ	Query SLOW speed limit
AC_AQ	Disc Query:		
AC_AQTYPE	$58	READ	Query media coding
AC_AQSTAT	$59	READ	Program status request
AC_AQUSER	$5A	READ	User code request
AC_AQENDM	$5B	WRITE	Query End of Search Phase
AC_AA	Audio Codes:		
AC_AABEEP	$60	ARGSXY	Sound Insert (beep)
AC_AAMUTE	$61	ARGSX	Audio mute over non-play phase
AC_AACX	$62	ARGSX	CX On/Off
AC_AA1ON	$63	WRITE	Audio 1 ON
AC_AA1OFF	$64	WRITE	Audio 1 OFF
AC_AA2ON	$65	WRITE	Audio 2 ON
AC_AA2OFF	$66	WRITE	Audio 2 OFF
AC_AAUDEX	$67	ARGSX	Audio Internal/External
AC_AV	Video Codes:		
AC_AVPION	$68	WRITE	Picture Index ON
AC_AVPIOF	$69	WRITE	Picture Index OFF
AC_AVCION	$6A	WRITE	Chapter Index ON
AC_AVCIOF	$6B	WRITE	Chapter Index OFF
AC_AVVION	$6C	WRITE	Video ON
AC_AVVIOF	$6D	WRITE	Video OFF
AC_AVOVLY	$6E	ARGSXY	Set/Read Overlay processor
AC_AVTXT	$6F	ARGSX	Teletext ON/OFF
AC_AF	Frame Codes		
AC_AFREQ	$70	READ	Frame code request
AC_AFINFO	$71	ARGSX	Load INFO register
AC_AFSTOP	$72	ARGSX	Load STOP register
AC_AFGOST	$73	ARGSX	Goto FRAME and still

AC_AFGOPL	$74	ARGSX	Goto FRAME and play
AC_AFGOCN	$75	ARGSX	Goto FRAME and continue
AC_AFRETI	$76	WRITE	Wait until FRAME INFO clears
AC_AFRETS	$77	WRITE	Wait until FRAME STOP clears
AC_AC	Chapter Codes:		
AC_ACREQ	$78	READ	Chapter request
AC_ACGOST	$79	ARGSX	Goto CHApter and still
AC_ACGOSPL	$7A	ARGSX	Goto CHAPTER and play
AC_AT	Time Codes:		
AC_ATREQ	$7C	READ	Time Code Request
AC_ATINFO	$7F	ARGSX	Load Time code INFO register
AC_ATGOPL	$7E	ARGSX	Goto time and play (CLV)
AC_ATRETI	$7D	WRITE	Wait until TIME INFO clears
AC_AS	Speed Codes:		
AC_ASFAST	$80	ARGSX	Set Fast speed modifier
AC_ASSLOW	$81	ARGSX	Set Slow speed modifier
AC_ASSTIL	$82	WRITE	Freeze video
AC_ASSTOP	$83	WRITE	Stop + mute video
AC_AJ	Jump Codes:		
AC_AJUMPF	$86	ARGSX	Instant Jump Fwd
AC_AJUMPR	$87	ARGSX	Instant Jump Rev
AC_AJUMPQ	$88	READ	Query jumping capability
AC_AJPFJF	$89	ARGSXY	Play Fwd Jump Fwd
AC_AJPFJR	$8A	ARGSXY	Play Fwd Jump Rev
AC_AJPRJF	$8B	ARGSXY	Play Rev Jump Fwd
AC_AJPRJR	$8C	ARGSXY	Play Rev Jump Rev
AC_AJHTJF	$8D	ARGSXY	Halt & Jump Fwd
AC_AJHTJR	$8E	ARGSXY	Halt & Jump Rev

AC_AX	Scan Codes:		
AC_AXCFWD	$90	WRITE	Scan forward once
AC_AXCRVS	$91	WRITE	Scan reverse once
AC_AM	Motion Codes:		
AC_AMGO	$92	WRITE	Play from media start
AC_AMPNF	$93	WRITE	Play forward NORMAL
AC_AMPNR	$94	WRITE	Play reverse NORMAL
AC_AMPSF	$95	WRITE	Play forward SLOW
AC_AMPSR	$96	WRITE	Play reverse SLOW
AC_AMPFF	$97	WRITE	Play forward FAST
AC_AMPFR	$98	WRITE	Play reverse FAST
AC_AMSTF	$99	WRITE	Step forward 1 frame
AC_AMSTR	$9A	WRITE	Step backward 1 frame

Index

Appendix E: Hardware and Software Companies

The following companies sell Amiga peripherals and/or software that you may wish to use. This list is by no means exhaustive. Consult Amiga-specific magazines and bulletin boards or your local Commodore dealer for more detailed information.

Graphics

Product	Company
Aegis Images	OXXI/Aegis Development
DeluxePaint III	Electronic Arts
Deluxe Photolab	Electronic Arts
Diamond	Impulse
Digi-Paint 3	NewTek
Express Paint 3.0	OXXI, Inc.
Photon Paint 2.0	MicroIllusions
Spritz	OXXI, Inc.

Digitizers

Digi-View 4.0	NewTek
Framegrabber	Progressive Peripherals & Software
Perfect Vision	SunRize Industries
Professional ScanLab	ASDG Inc.

Music

Deluxe Music	Electronic Arts
Instant Music	Electronic Arts
Synthia	The Other Guys

Digitized Sound

FutureSound	Applied Visions
Master Sound	Michtron
Quasar Sound	Centaur Software, Inc.
Sound Scape	Mimetics Corp.

Animation

ANIMagic	OXXI/Aegis
Animation: Editor	Hash Enterprises
Animation Station	Progressive Peripherals
DeluxePaint III	Electronic Arts
Digimate III	Mindware International
Photon Paint 2.0	MicroIllusions

Video

A2300 Genlock	Commodore
AmiGen	Mimetics Corp.
4004/4004S	Magni

Touch Screens

Elographic Touch Screen	Elographics
Future Touch	Amigo Business Computers
MicroTouch	MicroTouch Systems, Inc.

Company Addresses

Amigo Business Computers, Inc.
192 Laurel Road
East Northport, NY 11731
(516) 757-7334

Applied Visions
1 Kendall Square
Suite 2200
Cambridge, MA 02139
(617) 494-5417

ASDG, Inc.
925 Stewart St.
Madison, WI 53713
(608) 273-6585

Centaur Software
PO Box 4400
Redondo Beach, CA 90278
(213) 542-2226

Electronic Arts
1820 Gateway Dr.
San Mateo, CA 94404
(415) 571-7171

Elographics
105 Randolph Road
Oak Ridge, TN 37830
(615) 482-4100

Hash Enterprises
2800 E. Evergreen Blvd.
Vancouver, WA 98661
(206) 693-7443

Impulse Inc.
6870 Shingle Creek Pkway #112
Minneapolis, MN 55430
(612) 566-0221
(800) 328-0184

Magni Systems, Inc.
9500 S.W. Gemini Drive
Beaverton, OR 97005
(503) 626-8400

Michtron
576 South Telegraph
Pontiac, MI 48053
(315) 334-5700

MicroIllusions
17408 Chatsworth St.
Granada Hills, CA 91344
(818) 360-3715

MicroTouch Systems, Inc.
55 Jonspin Road
Wilmington, MA 01887
(508) 694-9900

Mimetics Corporation
PO Box 1560
Cupertino, CA 95015
(408) 741-0117

Mindware International
110 Dunlop W
Box 22158
Barrie, Ontario, L4M5R3 Canada
(705) 737-5998

NewTek
115 West Crane St.
Topeka, KS 66603
(913) 354-1146
(800) 843-8934

OXXI, Inc.
OXXI/Aegis Development
1339 E. 28th St.
Long Beach, CA 90806
(213) 427-1227

Progressive Peripherals & Software
464 Kalamath St.
Denver, CO 80204
(303) 825-4144

SunRize Industries
PO Box 1453
College Station, TX 77841
(409) 846-1311

The Other Guys
55 N. Main
Suite 301D
Logan, UT 84321
(801) 753-7620

Glossary

Glossary

A

AmigaVision

The program that allows you to create interactive, multimedia presentations and courses.

ANIM

The Aegis/Sparta compression format used by programs such as Deluxe Paint III and ANIMagic. ANIM files consist of a full starting frame. Each subsequent frame consists only of the differences between the frames, allowing for high compression of video. AmigaVision uses ANIM OPT 5, the latest version of ANIM, which allows for palette changing and timing control for each frame, a change over the older ANIM OPT 3. Elan Performer is an inexpensive package for converting older ANIM files to the new standard.

ASCII

Stands for American Standard Code for Information Interchange; a standard that controls communication between computers. Allows you to exchange files from one program to another.

B

Brush

A small picture created in any of the drawing programs, such as DeluxePaint. It may be incorporated into a screen in AmigaVision.

C

Child

An icon placed in the presentation outline below and to the right of a parent icon.

Chip RAM

The type of random access memory addressable by the Amiga custom chips. Chip RAM controls the amount of graphical information that can be displayed simultaneously, i.e., windows, screens, icons, etc.

Click (clicking on)

Moving the mouse pointer to an object on the screen and then quickly pressing and releasing the left mouse button. In AmigaVision a click is used to select an icon or object, or to activate a window.

Close Window Gadget

The small square at the top left corner of most requesters and windows. When clicked, it closes the present window.

Condition

A statement created in the Expression Editor which is equal to a true or false value.

Content Window

A window used for storing audio and visual files for use in projects. The Content Window is used to create commonly used sections, such as company logos or end credits, that can be copied into every Flow Outline you design. The Content Window is not executable.

Control

The direction or flow of the presentation or course. Control icons guide the order of items.

Cookie Cut

Means that when a brush is placed in the Object Editor, only the brush itself will be visible, not the background rectangle.

D

Delivery System

The computer hardware required to present the presentation or course in its final form.

Depth Gadget

A gadget in the upper right corner of a window that moves the window either to the front of the screen or behind all other windows on the screen.

Development System

The computer hardware required to create presentations and courses.

Digitized Sound

Recorded sound or sound effects that have been digitized and saved in computer files and can be played back. To use in AmigaVision, the files must be in 8SVX format. 8SVX is also called IFF ONE SHOT by some audio applications because the sound file represents a single octave, not an instrument. AmigaVision supports the multi-octave IFF instrument format as well, and loading such a file as a digitized sound will cause AmigaVision to play each octave. AudioMaster II is an excellent product for converting digitized sound into different formats, as well as for editing waveforms.

Digitizer

A device for bringing sounds and images into the computer for manipulation. An audio digitizer, such as Future, allows you to add real sounds and voices to your application. A video digitizer, such as Digi-View Gold, allows you to capture pictures and objects for use as graphical backgrounds and brushes. A frame grabber, such as FrameGrabber, allows you to capture moving images from video tape or a video camera. A special effects board, Live!, also offers the ability to capture multiple frames of video for animation.

Double Click

Rapidly pressing and releasing the mouse button twice. Double clicking on an icon in AmigaVision is a shortcut for opening its requester.

Drag

Moving an icon, window, or screen across the display. For instance, to move the icons in AmigaVision, click on one, and, holding the left mouse button down, drag the icon to its new location and release the button.

Drag Bar

The lined area at the top of most requesters and windows. It is used to drag the window to a new position.

E

Expression

In AmigaVision, a statement created in the Expression Editor which can be evaluated and results in a value. For instance: Score = Correct + Bonus, or Coord = X * sqrt(Y)

Expression Editor

Allows you to create variables, functions, and expressions for evaluation in the presentation or course.

F

Fast RAM

Random access memory which is not addressable by Amiga custom chips. It is used for storing programs and data.

Feedback

What the program does as a result of input. Feedback for a hit box may be a change in color or sound.

Field

An area created in the Object Editor used to define the screen position and format for text input.

Flow

Describes the direction and sequence of an AmigaVision presentation.

Flow Editor

The editing environment used for placing icons in the Flow Window and defining the icons through requesters.

Flow Window

The window in which you create the outline for the final project. This window automatically opens when you enter AmigaVision. More than one Flow Window may be open, as available memory permits.

Font

Defines the typeface used on the screen. In AmigaVision, font specifications include type and size as well as styles, such as boldface, italics, and underline. AmigaVision supports all standard Amiga bitmap fonts, including Color Fonts, if the external ColorText program is running.

Function

System-supplied routines which return current values of system settings. For instance, the response() function returns the response string of the most recently clicked hit box.

G

Gadget

An area in a window, requester or screen that allows you to change what's being displayed or to communicate with the Amiga.

H

Hardware

Consists of the computer, monitor, printer and other tangible items.

Hit Mask

The name for brushes that have been defined as hit boxes. The user must click directly on the picture and not on the background to activate this hit box.

Hit Box

An area of the screen defined in the Object Editor. A hit box can be clicked on by the mouse or touched. A response string can be associated with each hit box.

I

Icon

A small picture that represents a specific function in AmigaVision. It can be moved with the mouse and placed into the presentation outline.

Icon Menu

Icons that appear at the bottom of the AmigaVision screen. Each contains a submenu. When you click on the icon, you will view the submenu. These icons cannot be placed into the outline.

Icon Submenu

Icons of a particular type contained under each main menu icon. They may be placed into the presentation outline.

IFF

A standard file format for the Amiga. IFF stands for Interchange File Format and was developed by Electronic Arts and Commodore-Amiga to make it easy to manipulate files by multiple products. Each IFF file contains a four letter identifier that marks its type: ILBM, graphical bitmap; ANIM, Aegis/SPARTA animation; SMUS, simple music format; and 8SVX, one-shot digitized sound. AmigaVision files are in IFF format with the identifier AVCF. Most files created for use on the Amiga will be in IFF format. Refer to the manuals for the specific programs to be sure.

Index

The ability to view the frame number of the video being played. If index is on, the frame number will be displayed while the video is playing.

Interrupt

An interruption in the presentation that is activated by some input from the user. When activated, the interrupt's children will be executed before the presentation continues.

Interlace

A technique for refreshing the display screen by alternately displaying odd and even lines. In AmigaVision, describes a standard screen resolution mode.

L

Level

The vertical column in which the icon appears. The leftmost column is the highest level.

Loop

A group of several icons to be executed more than once.

M

Menu

A list of the choices you can select from in a program. Menus are accessed by pressing and holding the right mouse button. They are located on the main AmigaVision screen and in the Object Editor. If you release the button when one of the menu items is highlighted, that function is selected.

Module

An icon which can contain any other icon as a child. Used to contain the entire course or presentation or to organize a presentation into smaller sections.

Mouse

The small, palm-sized device used to control the pointer and to communicate with your Amiga. It contains two buttons for selecting icons, dragging icons, windows or screens, and displaying menus.

O

Object Editor

Allows you to create display objects (geometric shapes and brush images) and hit boxes to place on the screen either alone or on top of a picture or video frame.

Outline

The icons which are placed into the Flow Window and specify the sequence of events in the presentation.

Overscan

A display format where there are no visible borders on the monitor; also called full video. This graphic mode is available in most art and animation packages and allows you to make use of the entire monitor.

P

Palette

A set of colors for a screen. Can be accessed through the Object Editor and the Screen Palette submenu item on the pull-down menu and Screen Definition requester.

Parent

An icon that can contain children icons. The parent will appear above and to the left of its group of children. Parent icons have a small hollow triangle in the lower right corner. When children are placed under the icon, the triangle becomes solid.

Partner

An icon which is placed directly beside another icon. Only certain control icons require a partner, but almost any icon can be a partner.

Polygon

A shape created in the Object Editor. It has unlimited sides and no uniform shape.

Present . . .

The AmigaVision option you select to run a presentation.

Preview

A gadget on requesters that lets you see an operation without running the program or exiting the requester.

Project

Refers to the course or presentation while you are creating it, as well as to the finished product.

R

Requester

A window that opens when certain menu items are selected or gadgets are clicked on. You use this window to define settings in AmigaVision. Requesters are also used to inform you of certain events or to give warnings.

S

Scroll Window

A window with a bar along its right hand side. The bar contains up and down arrows which, when clicked on, will scroll the contents of the window.

Select

To click on an object to activate it. For instance, if multiple windows are open on one screen, only one can be active (ready for interaction from you). When you click on an AmigaVision icon, a square area behind it turns black to indicate its selected state.

Sibling

An icon placed directly above or below another icon, on the same level in the outline.

Sizing Gadget

A gadget in the lower right corner of a window that allows you to change the size of the windows.

Slider Gadget

A gadget containing a cursor which indicates how fast or slow (high or low) the value is. The cursor may be dragged to change the value.

SMUS

Simple Musical Score files containing music and created in most music programs. Sonix, a program which also creates SMUS files, does not use IFF standard instrument format, so such files must be edited using Deluxe Music to access IFF sampled instruments.

Subroutine

A group of icons which contain a small program.

Synthesized Speech

Written text (ASCII format) which is spoken by the computer.

T

Telescope

An option on the Edit pull-down menu that lets you condense child icons into a parent to conserve space in the Flow Window.

Toggle

A feature which operates like a switch. When the Toggle is on and a hit box is clicked on, the box will change and remain changed until it is clicked on again.

Transition

Describes the way one image replaces another on the display. Access the transition options by clicking on the Transition gadget in an icon requester.

V

Variable

Provides temporary storage for values. Variables are created in the Expression Editor.

Videodisc Controller

Allows interactive control of the videodisc player. You may view the video and save frame numbers and commands.

W

Window

A framed area of the screen that can contain a program or file. The most often-used windows in AmigaVision are the Flow and Content Windows.

Workbench

The Amiga's icon-based [graphical] user interface software.

Index

Index